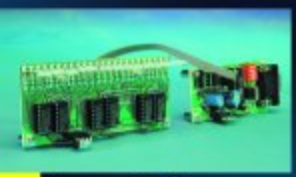
MARCH 2003
£3.45

ELEKTOR ELECTRONICS

THE ELECTRONICS & COMPUTER MAGAZINE

www.elektor-electronics.co.uk



Two-Colour Running Light

Guitar Effects Switch

COM Port Tester

Serial Port Driver for Windows

Intelligent Fan Timer

R/C Model Switcher

AVRee Development System

introducing the AT90S2313 micro

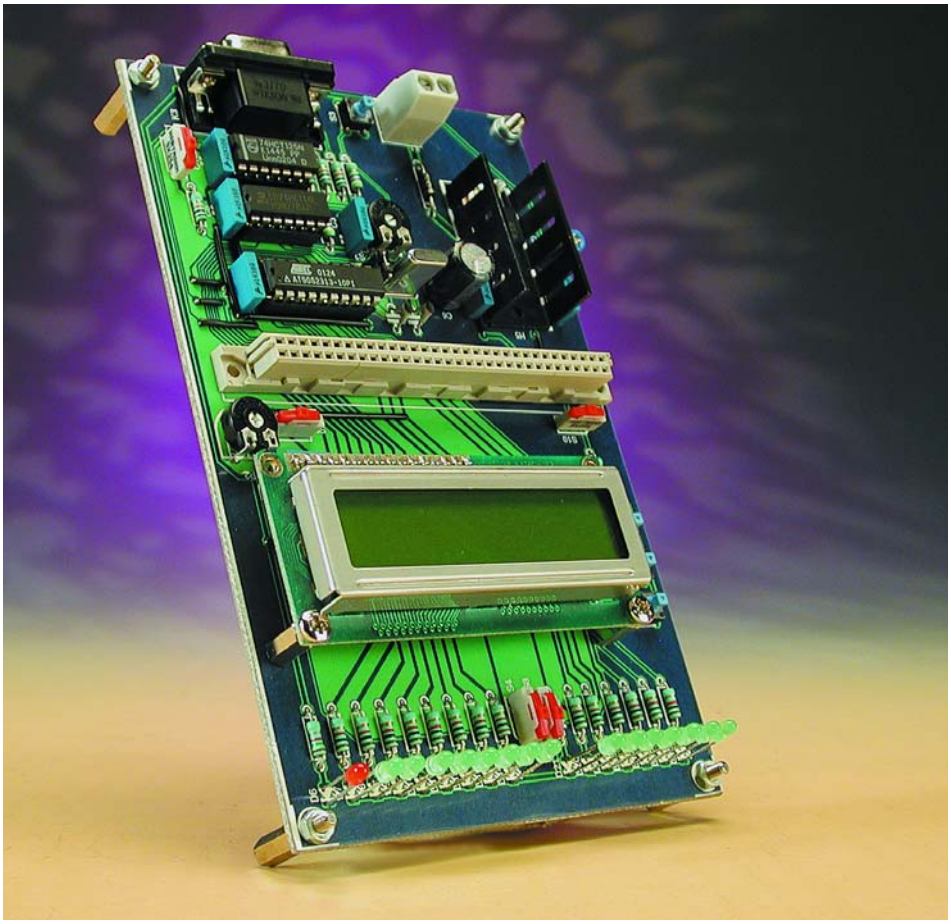
03
9 770268 451104

AVRee Development System

Using the AT90S2313

Design by A. Schumacher and R. Behl

After our publications covering the 89S8252 Flash Micro Board (December 2001) and the PICee Development System (February 2002) it's high time to devote some attention to the AT90S2313 microcontroller. As with the other two designs we describe a powerful single-board computer that's not just excellent for educational purposes but also for general use by microcontroller enthusiasts.



The development system described in this article is the spitting image of the PICee board you may have seen in the February 2002 magazine. This is not surprising, for two reasons. Firstly, the AT90S2313 resembles the PIC16F84 in that it has two supply connections, a reset pin, two connections for the oscillator circuit and a number of port pins that allow the micro to be linked to external hardware. Just like its predecessor, the present development system features a number of LEDs for easy visualisation of port pin logic states during software development, a number of pushbuttons, one connector for the link to a standard alphanumerical LC display module and another for external hardware. Also, the AVRee Development System is yet another design submitted to us by students at the Ludwig Geissler School of Hanau, Germany, so it's less than surprising to see a great deal of similarity with the (hugely successful) PICee board.

The AT90S2313 is a member of the renowned 'AVR' microcontroller family produced by Atmel. The most

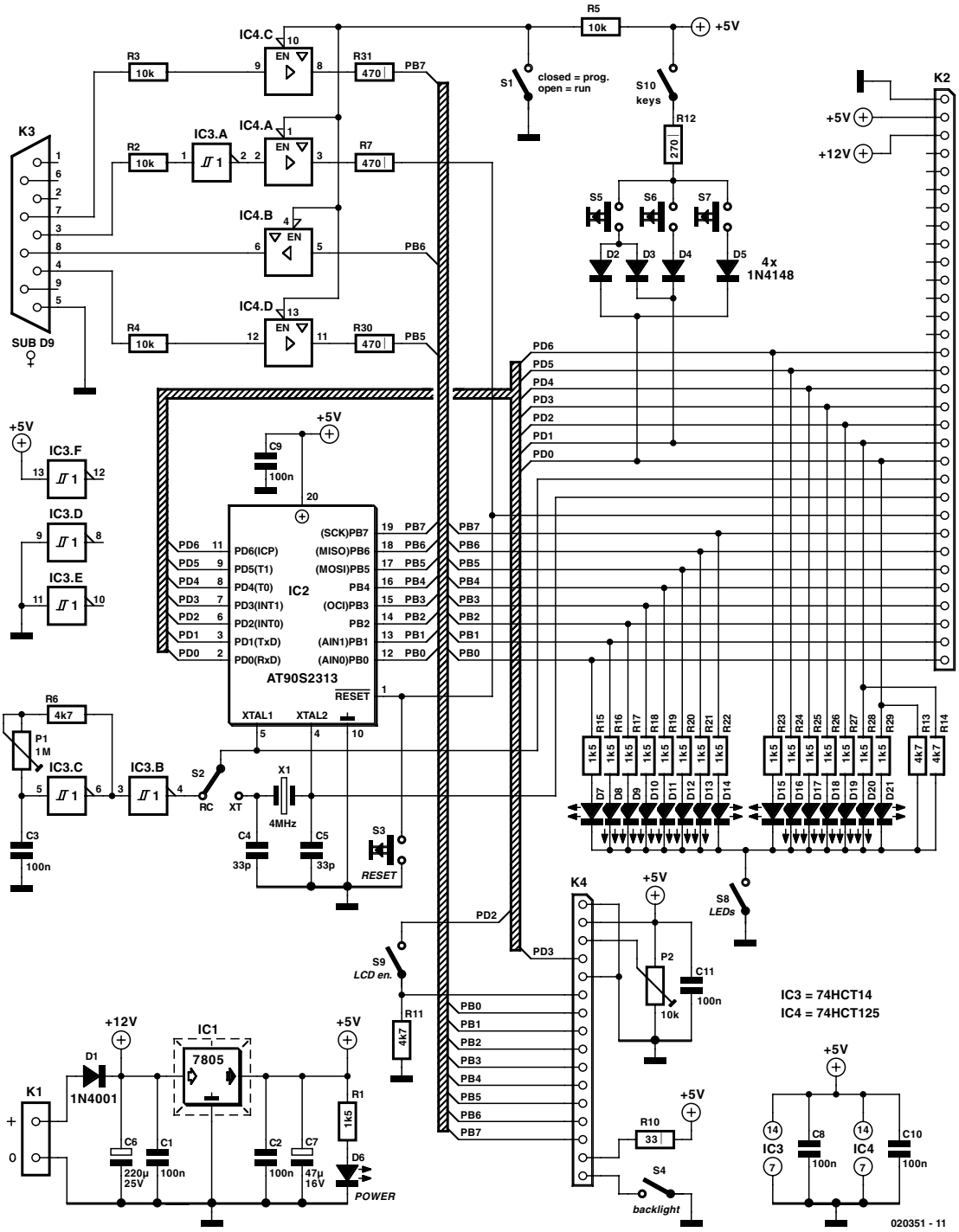


Figure 1. The general structure of the circuit bears great resemblance to that of our successful PICee system.

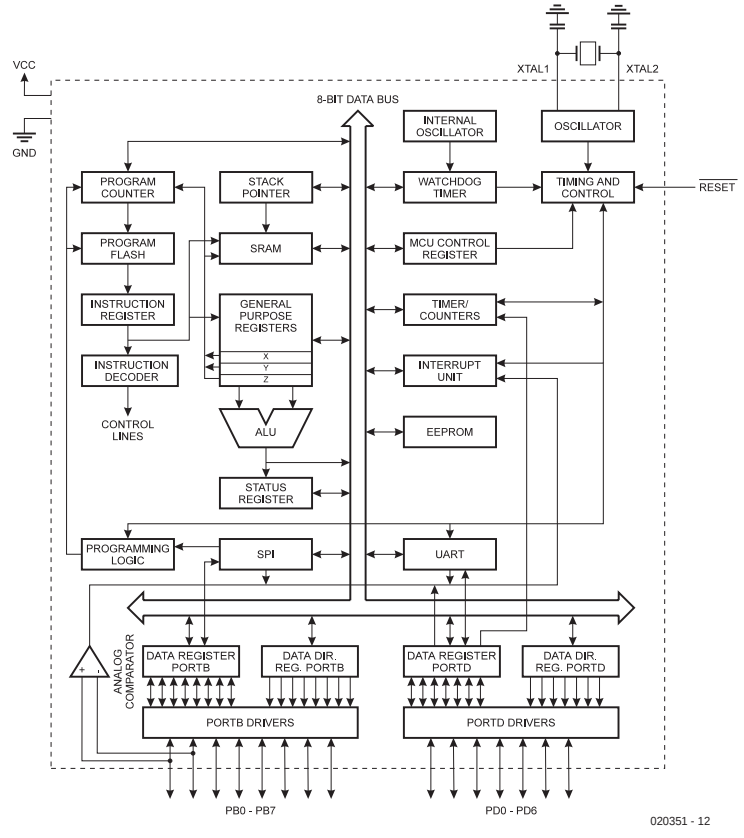
salient data of this chip are given in an inset. Just like all other modern microcontrollers the devices in the AVR series sport an internal Flash program memory which means goodbye to the cumbersome and tedious process of programming (internal) EPROM using a special programmer and subsequent erasing in a separate EPROM eraser unit. From now on, some simple add-on hardware on the development system and a serial link with the PC running free software are the ingredients for quick and easy software development and testing.

Hardware

The circuit diagram of the development system is shown in **Figure 1**. The board may be powered from a dead standard mains adapter (a.k.a. battery eliminator) capable of supplying a direct voltage of between 9 V and 15 V. The current requirement on the adapter will largely depend on the hardware connected to the circuit. Normally, a 500-mA adapter will be more than sufficient. It should be noted, however, that any backlight lamp(s), especially in older LCD modules, may draw currents of several hundred milliamps. To make sure the voltage regular does not become overheated during

AT90S2313 Features

- Utilizes the AVR® RISC Architecture
- AVR – High-performance and Low-power RISC Architecture
  - 118 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Up to 10 MIPS Throughput at 10 MHz
- Data and Non-volatile Program Memory
  - 2K Bytes of In-System Programmable Flash
    Endurance 1,000 Write/Erase Cycles
  - 128 Bytes of SRAM
  - 128 Bytes of In-System Programmable EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - Programming Lock for Flash Program and EEPROM Data Security
- Peripheral Features
  - One 8-bit Timer/Counter with Separate Prescaler
  - One 16-bit Timer/Counter with Separate Prescaler, Compare,
    Capture Modes and 8-, 9-, or 10-bit PWM
  - On-chip Analog Comparator
  - Programmable Watchdog Timer with On-chip Oscillator
  - SPI Serial Interface for In-System Programming
  - Full Duplex UART
- Special Microcontroller Features – Low-power Idle and Power-down Modes
  – External and Internal Interrupt Sources
- Specifications – Low-power, High-speed CMOS Process Technology
  – Fully Static Operation
- Power Consumption at 4 MHz, 3V, 25°C – Active: 2.8 mA
  – Idle Mode: 0.8 mA
  – Power-down Mode: < 1 µA
- I/O and Packages – 15 Programmable I/O Lines
  – 20-pin PDIP and SOIC
- Operating Voltages – 2.7 - 6.0V (AT90S2313-4)
  – 4.0 - 6.0V (AT90S2313-10)
- Speed Grades – 0 - 4 MHz (AT90S2313-4)
  – 0 - 10 MHz (AT90S2313-10)



regular use of the system, the circuit board allows for a standard U-shaped heatsink (e.g., Fischer type ICK35) to be fitted. As is customary, the circuit includes a power diode, D1, for protection against damage caused by accidental supply voltage reversal. LED D6 lights when the circuit receives the proper supply voltage.

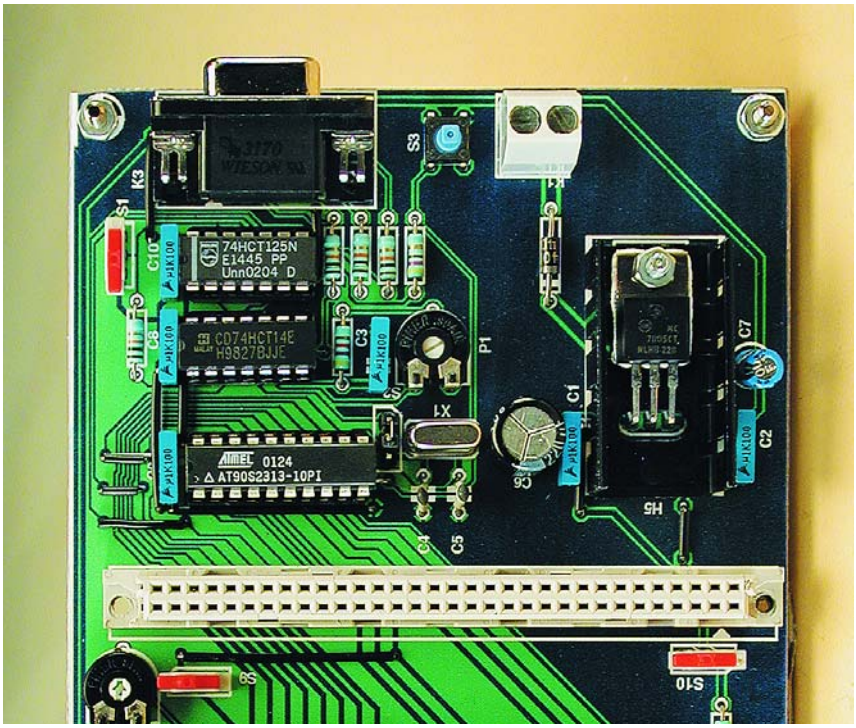
A quad tristate buffer, IC4, allows the on-board microcontroller to be programmed. The buffer effectively copies signals from the PC's serial port to the microcontroller. By way of three port pins (MOSI, MISO and SCK) and the microcontroller reset circuit (IC2), the programmer software that runs on the PC is capable of erasing, reading and programming the Flash memory inside the AT90S2313. This is possible provided switch S1 is closed so that the relevant signals are actually passed. If the switch is open, the buffer outputs are at high-impedance, causing the programmer interface to be 'disconnected' from the rest of the circuit. Resistor R7 prevents the output of buffer IC4a being short-circuited when the interface is switched on and reset switch S3 is accidentally pressed. A similar short-circuit protection is implemented on IC4d and IC4c by resistors R30 and R31.

Note that the AVR controller does not require a separate programming voltage. You may recall that the PICee board had a dedicated step-up converter circuit for the 13.5 V programming voltage. By contrast, the AVR is happy with just 5 V for erasing and programming.

Regarding the reset circuit, dyed-in-the-wool microcontroller fans will not have failed to notice that there is no external power-on reset circuit on the reset pin of the AT90S2313. The usual R-C combination, so familiar from other controllers, is not necessary here. The reset circuit inside the AT90S2313 has a pull-up resistor, while on-chip circuitry looks after the power-on reset. In normal use, this pin may be left unconnected.

Pushbutton S3 has been added to enable users to manually reset the microcontroller.

Looking at the oscillator circuit, switch S2 enables you to choose between a continuously adjustable R-C oscillator (IC3c, R6, P1 and C3) and a standard quartz crystal oscillator of



the 'Pierce' variety consisting of X1, C4 and C5. It is also possible to apply an external clock signal to the microcontroller by omitting the last three components and connecting a signal via XT1 on connector K2. Note, however, that this requires S1 to be set to the 'XT' position. IC3 (pin 4 is the oscillator output) may not be omitted just like that because IC3a forms part of the programmer interface.

Connector K4 provides the gateway to an (optional) standard alphanumerical LCD module. Preset P2 is for the LCD contrast adjustment. The module may only be driven if switch S9 is closed. If not, the LCD's enable signal is held low by R11, whereby communication is of course prevented. Also note that it is not possible to read back data from the display because R/$\overline{W}$ at pin 5 of K4 is tied to ground which rules out data being sent by the display

If the display is being driven by a program, the software has to allow enough time for the LCD to process its data. After all, it is not possible to read the status of the module using the BUSY line.

The LCD backlight, if used, may be powered via pins 15 and 16 of connector K4. The requisite current limiting is afforded by R4, while switch S4 allows the backlight to be switched on and off.

An array of 15 LEDs (D7-D21) is available to signal the logic state of the same number of AT90S2313 port pins as you develop and debug your software for the board. If desired, the LEDs may be disconnected from the controller using switch S8.

The development system also has three pushbuttons (S5, S6, S7) that may be used for any purpose requiring manual control in your programs, but only if S10 is closed.

Controller port line PD1 goes logic High if S6 is pressed, PD0 if S7 is pressed, while both lines are pulled High if S5 is closed. In this situation, resistors R13 and R14 act as pull-downs on PD0 and PD1.

Finally, there's DIN41612-style connector K2 for easy connection of external circuitry to the development system. This connecter carries all microcontroller signals and supply rails, allowing you to connect your own hardware extensions to the board. For example, using an adapter installed on K2, it becomes possible to program other controllers from the AVR-RISC series. Obviously, the controller on the AVRee board has to be removed in that case.

Printed circuit board

The PCB artwork shown in **Figure 2** looks very much like that published for the PICee system — which again is not surprising in view of the strong resemblance between the two 'platforms'. The board is divided in two if you like by extension connector K2. The lower area comprises most of the electronics, the upper area, the LCD, LEDs and most of the switches.

Here, too, we have succeeded in keeping the board **single-sided** which has a positive effect on the cost of the project. Of course, some wire links could not be avoided but their total number remains modest at just 15.

Speaking of wire links, they should be the first 'components' to install when you start stuffing the board. Moreover, some of the wire links are in tricky places on the board. Two, for example, are located under thje socket for the processor (IC2), another under the socket for IC3 and two run worryingly close beside C8 and C9. Believe it or not, wire links are the most frequently forgotten items on otherwise neatly populated boards we receive from the odd reader complaining that his/her project fails to work despite 'great efforts' at building it.

The number of components to fit is modest given the size of the board, so stuffing the board is not likely to cause headaches or even fatigue. Of course, you should work carefully and fit everything in accordance with the parts list and the component overlay.

If desired, jumpers may be used in lieu of toggle switches S1, S4, S8, S9 and S10. Actually, a jumper is to be preferred over a switch in the case of S2. Integrated circuit s IC2, IC3 and IC4 are preferably installed in sockets. The LCD may be plugged onto the board using SIL connectors and the heatsink for voltage regulator IC1 was already discussed.

Do observe the polarity of the diode, the electrolytic capacitors and the LEDs, and run a visual inspection on the completed board before applying the supply voltage.

Programming tools

A vast amount of literature is now available for those of you who want to learn the programming techniques specific to the Atmel microcontroller family. The Internet is, of course, the biggest resource to start looking. Even a brief search should produce masses of applications, hardware, software and other useful stuff. Not surprisingly, the majority of technical universities and polytechnics these days offer at least a gateway to information on these Atmel controllers.

A number of programming tools are downright essential if you want to develop your own programs. Among others we would recommend Atmels's own AVR STUDIO 4 (editor, assembler, simulator), as well as the

two programmers PonyProg2000 and ICPROG. These programmers run under Windows 95/98/NT/2000/ME/XP, and may be picked up as freeware from Internet websites. Atmel's own website provides all datasheets as well as a vast number of example programs (application notes). Incidentally, using PonyProg2000 it is also possible to program PICs and EEPROMs.

Once the source code of a program (*.asm) has been written and converted into machine code (*.hex), the programming software may be used to blow the latter file into the micro-

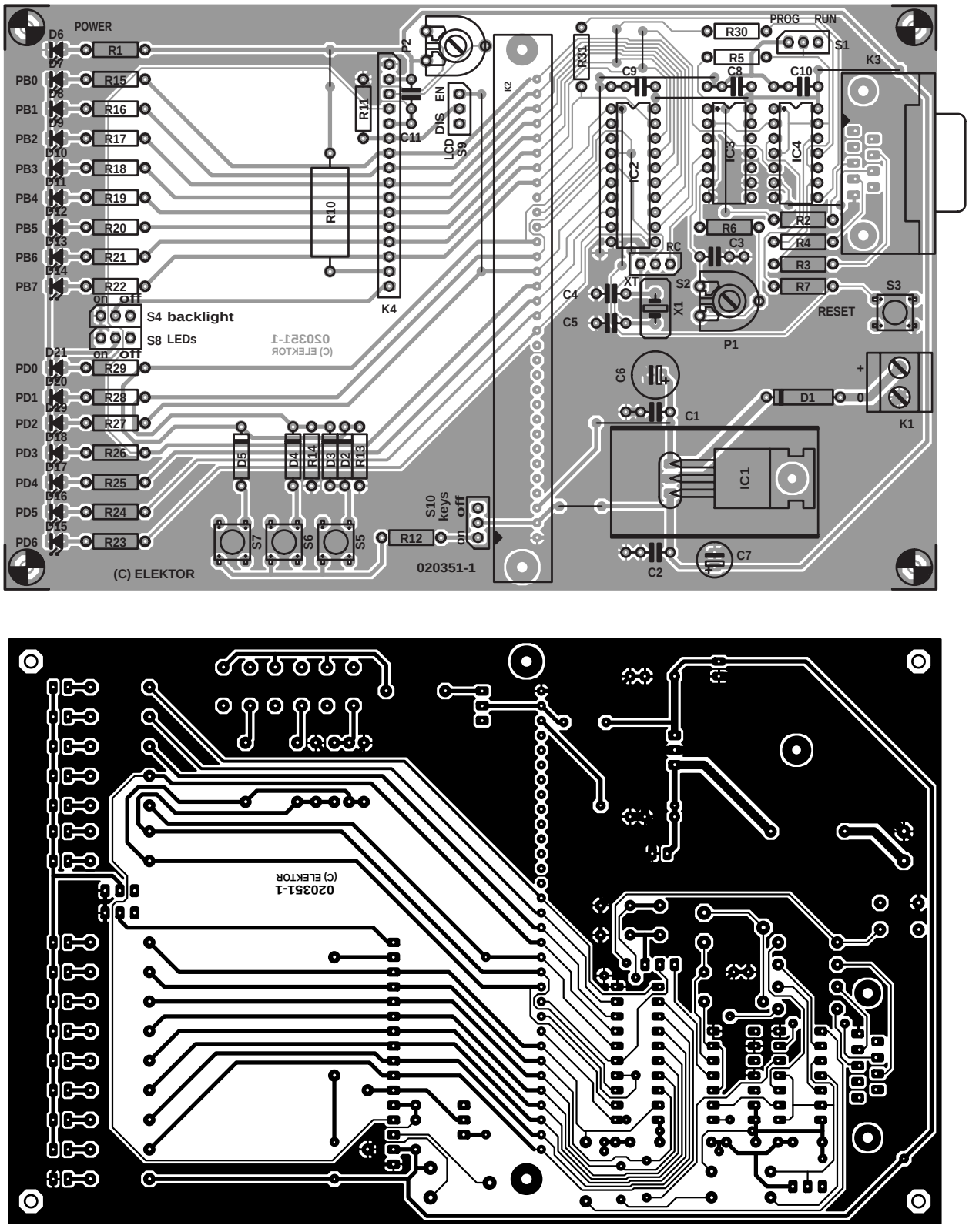
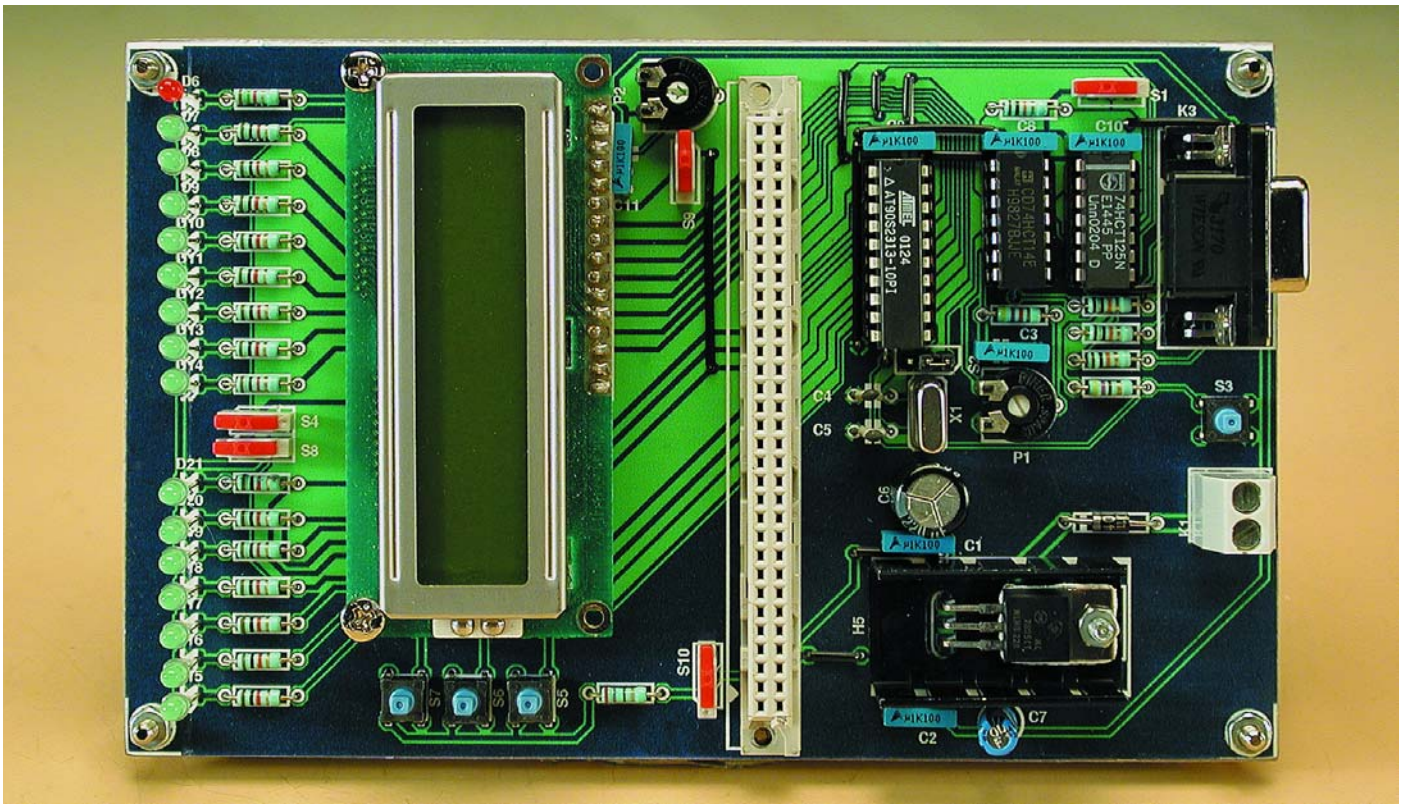


Figure 2. The PCB design may also look familiar to insiders. The number of wire links on the single-sided board is just 15.

controller's Flash memory. The Flash memory may be erased many times. Those of you who prefer higher programming languages will easily find several compilers on the Internet, including BASIC, C and Pascal.

To download machine code to the board, connect it to the serial port (COM) of your PC and switch S1 to the 'programming' position. Once the program has been downloaded, S1 is switched back and the controller is reset by pressing S3. That's all there is to it, no need to unplug the programming cable!

(020351-1)



COMPONENTS LIST

Resistors:
R1,R15-R29 = 1kΩ5
R2-R5 = 10kΩ
R6,R11,R13,R14 = 4kΩ7
R7,R30,R31 = 470Ω
R10 = 33Ω
R12 = 270Ω
P1 = 1MΩ preset
P2 = 10kΩ preset

Capacitors:
C1,C2,C3,C8-C11 = 100nF
C4,C5 = 33pF
C6 = 220µF 25V radial
C7 = 47µF 16V radial

Semiconductors:
D1 = 1N4001
D2-D5 = 1N4148
D6 = LED, yellow
D7-D10,D15-D21 = LED, green
D11-D14 = LED, red
IC1 = 7805
IC2 = AT90S2313
IC3 = 74HCT14
IC4 = 74HCT125

Miscellaneous:
K1 = 2-way PCB terminal, lead pitch 5mm
K2 = DIN41612 socket (female), B model, (Conrad Electronics # 741582)
K3 = 9-way sub-D socket, angled pins, PCB mount
K4 = 16-way SIL header
S1,S2,S4,S8,S9,S10 = jumper or switch, Hartmann type SX254 (Conrad Electronics # 708062)
S3,S5,S6,S7 = miniature pushbutton (Conrad Electronics # 700460)
X1 = 4-10 MHz quartz crystal with socket (see text)
Heatsink for IC1
LCD, 2 x 16 characters, e.g., Displaytech 162
PCB, order code **020351-1** (see Readers Services pages)
Disk, example programs, order code **020351-11** or free download

Web links

www.atmel.com/atmel/products/prod23.htm
www.avr-asm-tutorial.net
www.lancos.com/
www.ic-prog.com

The authors/designers of this project may be contacted by email:

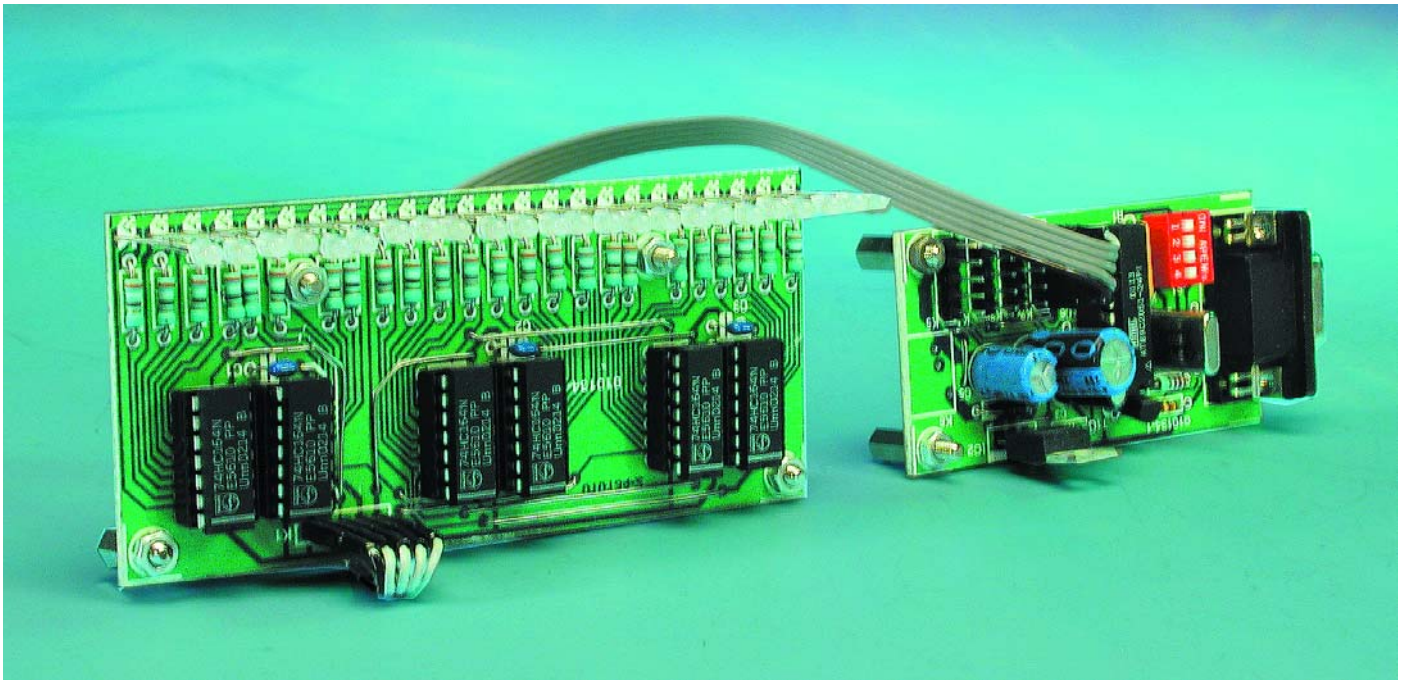
RalfBehl@softhome.net
AndreSchumacher@gmx.de

Two-Colour Running Light

with microprocessor control

Design by A. Köhler

Lighting effects are always popular and we frequently publish such projects in the pages of *Elektor Electronics*. Here is a circuit that uses a microcontroller to drive up to 96 bicolour LEDs in a wide range of patterns. A particular feature of the design is the way it exploits the persistence of human vision.



Instead of `regular' LEDs this running light uses so-called bicolour LEDs which can emit two different colours, usually red and green. These are available in versions with three connections (a common cathode and separate anodes) or with two connections. We use the second type in this circuit, where the two LEDs are connected in antiparallel so that one LED or the other lights according to the direction of current flow.

A special circuit is required to drive the bicolour LEDs. Since the LEDs generally only require a current of between 5 mA and 20 mA to light with sufficient brightness, the circuit does not have to switch a particularly large current. HCT logic devices, which can sink or source up to 8 mA, can therefore safely be used. The bicolour LEDs are connected

between two such outputs. If the two outputs are at the same logic level, both LEDs will remain dark; if the levels are different either the red or the green LED will light, according to the direction of current flow.

Since every LED will require two outputs, they cannot be driven directly from the microcontroller's outputs. A serial drive circuit is therefore used that requires just two signals: with just a data signal and a clock signal we can control a large number of LEDs. The serial-to-parallel conversions are done by HCT shift registers. In the first phase of operation the data bits are shifted rapidly through the shift registers: so rapidly, in fact, that this operation in not visible to the naked eye. In the second phase of operation the data remains at the outputs of the shift registers for a longer period and thus appears stable. After a brief pause a new pattern is loaded into the shift registers. With a suitable physical arrangement of the LEDs the effect can be improved still further.

The microcontroller...

The circuit consists of a central microcontroller unit and up to four bicolour LED modules. The arrangement shown in **Figure 1** is a straightforward microcontroller circuit employing an Atmel 89C2051. The data sheet for this device can be found at www.atmel.com/atmel/acrobat/doc1045.pdf. Components C3 and R4 form a power-on reset network. A primitive RS232 interface is provided via pin 2 for receiving data only; the transmit function via pin 3 is not implemented. Transistor T1 performs the voltage level conversion. D1 limits negative input voltages, while R3 limits the base current of T1 as well as the current through the diode. This circuit removes the need for the charge pump circuit that would otherwise be required.

The oscillator connected to pins 4 and 5 uses a 7.3728 MHz crystal from the junk box. This oscillator frequency restricts the choice of available RS232 baud rates, but simplifies the programming of longer delay loops. Any other crystal could of course be used, although a few delay constants in the software would then have to be adjusted to suit.

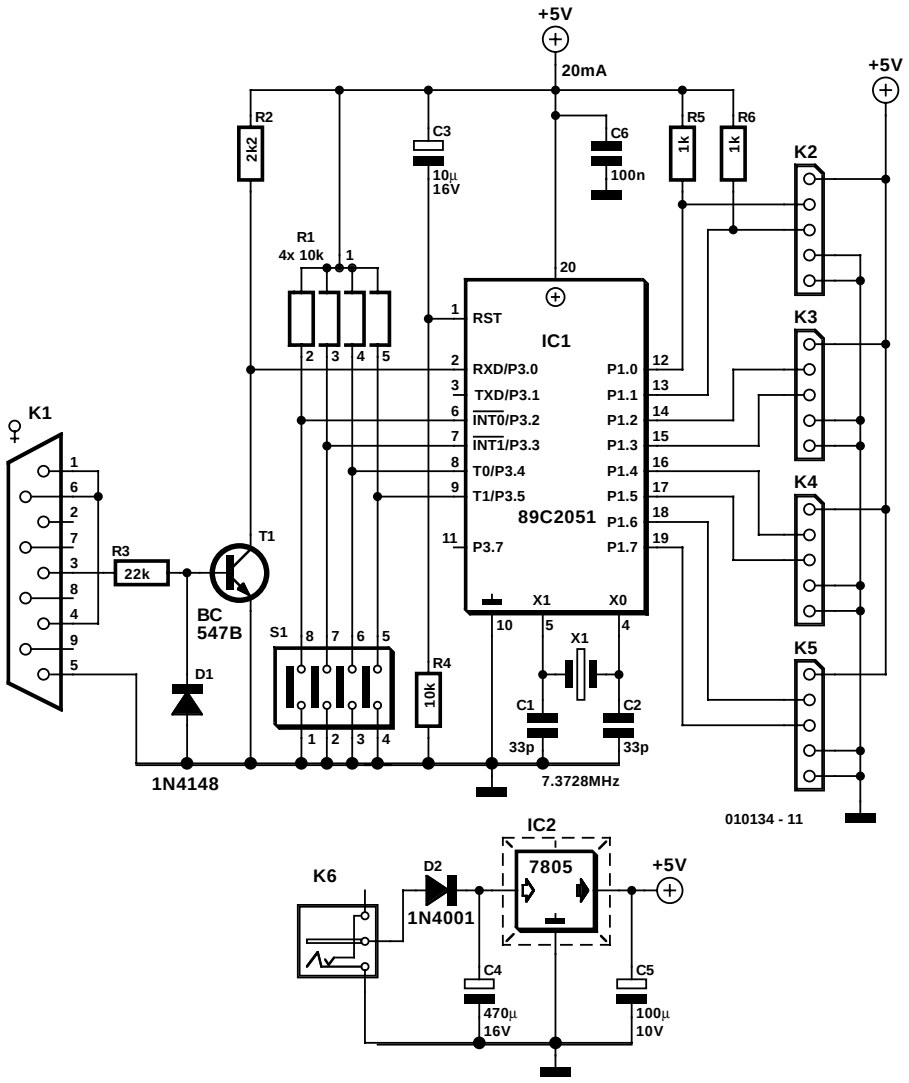


Figure 1. This minimal microcontroller circuit can control up to 96 bicolour LEDs.

Pins 6 to 9 are connected to a DIP switch with pull-up resistors. Depending on the software in the microcontroller, these can be used, for example, to select between light patterns, or to set the baud rate of the RS232 interface. Neither of these options is yet implemented in the software.

The LED modules are controlled via port 1. Each has a data signal (P1.0, P1.2, P1.4 and P1.6) and a clock signal (P1.1, P1.3, P1.5 and P1.7) to control a group of LEDs via their shift registers. Since the connections are short no protection circuitry is necessary. A point to note is the provision of pull-up resistors on port pins P1.0 and P1.1. These pins serve as non-inverting and inverting inputs (respectively) to an internal comparator. Without pull-up resistors of less than 2 kΩ the LEDs would not be driven correctly: the other pins of port 1 have internal pull-ups. The current drawn remains under the 15 able carrying data and clock as well as power.

This is derived from a mains power supply which must have an output voltage of at least 9 V and be able to deliver a current of 200 mA per LED module plus 20 mA for the controller board. That can add up to a sizeable total current, and so a heatsink must be fitted to voltage regulator IC2.

... and the four LED modules

As can be seen from **Figure 2** each LED module consists of 24 bicolour LEDs and six 74HCT164 shift registers. The data bits are presented to the data inputs (pins 1 and 2) of IC1, the first shift register, and then taken

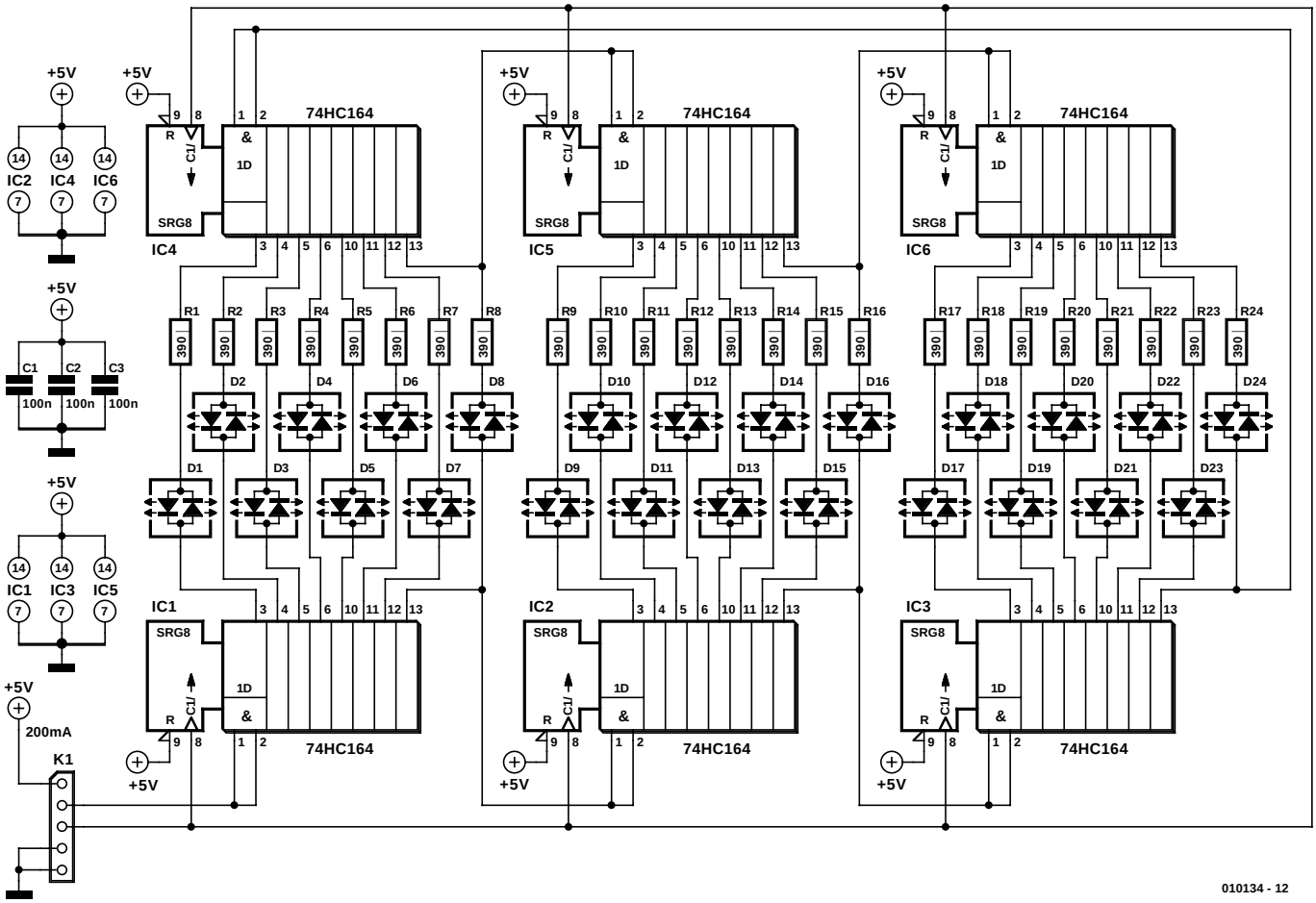


Figure 2. Two shift register outputs are used per bicolour LED.

from the final output (pin 13) to the inputs of the second shift register. In this way a chain is formed of the six shift registers, which are all clocked together (by pin 8).

The LEDs are connected to the outputs of the shift registers with series current-limiting resistors. A value of 390 Ω gives a current in the red LED of 9 mA, in the green of 7 mA. This is at the limit of what the logic ICs can supply, and does indeed give adequate brightness.

Populating the circuit boards should not present any difficulties, as long as the `smallest components first' principle is adhered to. Watch out for the three wire links. The microcontroller on the control circuit board (**Figure 3**) should be fitted in a socket. If connection to a PC will not be required, the sub-D connector and level shifting circuit can be dispensed with. Also, if the DIP switch will not be used, it too can be dispensed with, along with its pull-up resistors. Unused inputs should however be tied to a fixed voltage level, RxD to +5 V.

The single-sided printed circuit board for the LED module could not be routed without wire links, as clearly shown in **Figure 4**. Insulated wire should be used for the links, since in places they run close alongside one another, have bends or pass under IC sockets. Apart from that, there is little else to say about the circuit board.

Software variants

Now that we have seen that the hardware does not contain any big surprises, the key to the circuit must lie in the software. The software can be downloaded free of charge from the *Elektor Electronics* website (under this month's Free Downloads: look for number **010134-11**). The software is available not just as `ready to burn' hex code, but also as assembler source code.

The basic version of the software does not use the DIP switch. It could, for example, be used to select between a number of different patterns; this could also be done via the serial port.

The software is built around the following routines.

TAKT 1 to TAKT 4 (clock 1 to clock 4)
These routines generate the clock signals for the shift registers. In order to allow the modules to be updated at different times, each module is provided with its own routine.

MUAU 1 to MUAU 4
These routines emit the patterns, and are designed for LED modules with six shift registers, the number of shift registers being given in R2. R1 is used as a bit counter. A byte is fetched from the pattern table and split into individual bits via the carry flag. When a complete byte has been transmitted, the next byte is fetched from the table.

ZEIKO (time correction)
To allow all modules to use the same pattern, the pattern table pointer must be adjusted after each module has been processed.

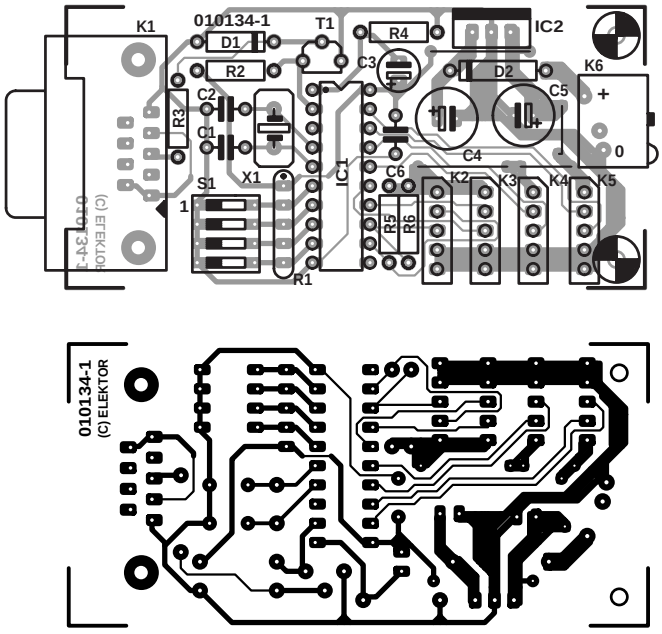
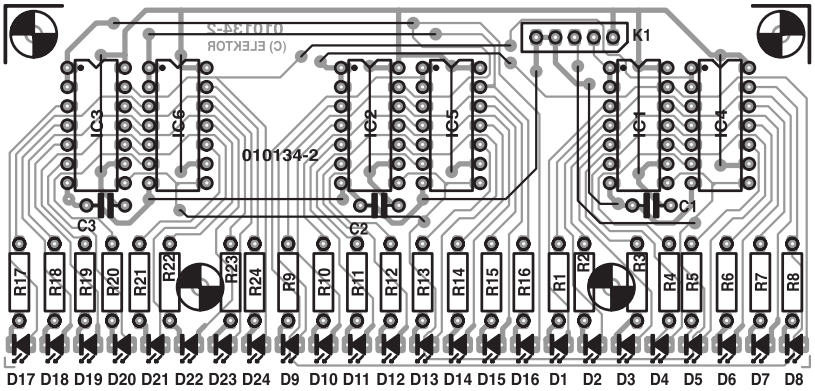


Figure 3. The microcontroller printed circuit board.

LOE1, LOE2 (clear 1, clear 2)
These routines shifts zeros into all the shift registers of one module, thus extinguishing all the LEDs.



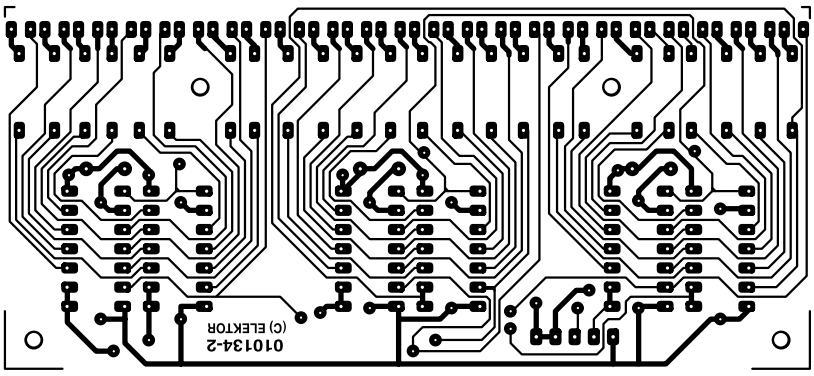


Figure 4. Beware! There are many wire links on the LED module printed circuit board.

COMPONENTS LIST
Controller board

Resistors:
R1 = 4-way 10kΩ SIL array
R2 = 2kΩ2
R3 = 22kΩ
R4 = 10kΩ
R5,R6 = 1kΩ

Capacitors:
C1,C2 = 33pF
C3 = 10µF 16V radial
C4 = 470µF 16V radial
C5 = 100µF 16V radial
C6 = 100nF

Semiconductors:
D1 = 1N4148
T1 = BC547B
IC1 = AT89C2051-12PC, programmed, order code **010134-41**
IC2 = 7805

Miscellaneous:
K1 = 9-way sub-D socket, angled pins, PCB mount
K2-K5 = 5-way SIL pinheader
K6 = mains adaptor socket, PCB mount
PCB, order code **010134-1**
Disk, order code **010134-11**

ZEIT (time)
This is a small delay loop that slows the display down to a reasonable speed. This speed can easily be adjusted.

The largest part of the program consists of the nine pattern tables. With the help of the documentation it should be possible to replace the contents of the tables with your own patterns.

(010134-1)

COMPONENTS LIST
LED board

Resistors:
R1-R24 = 390Ω

Capacitors:
C1,C2,C3 = 100nF

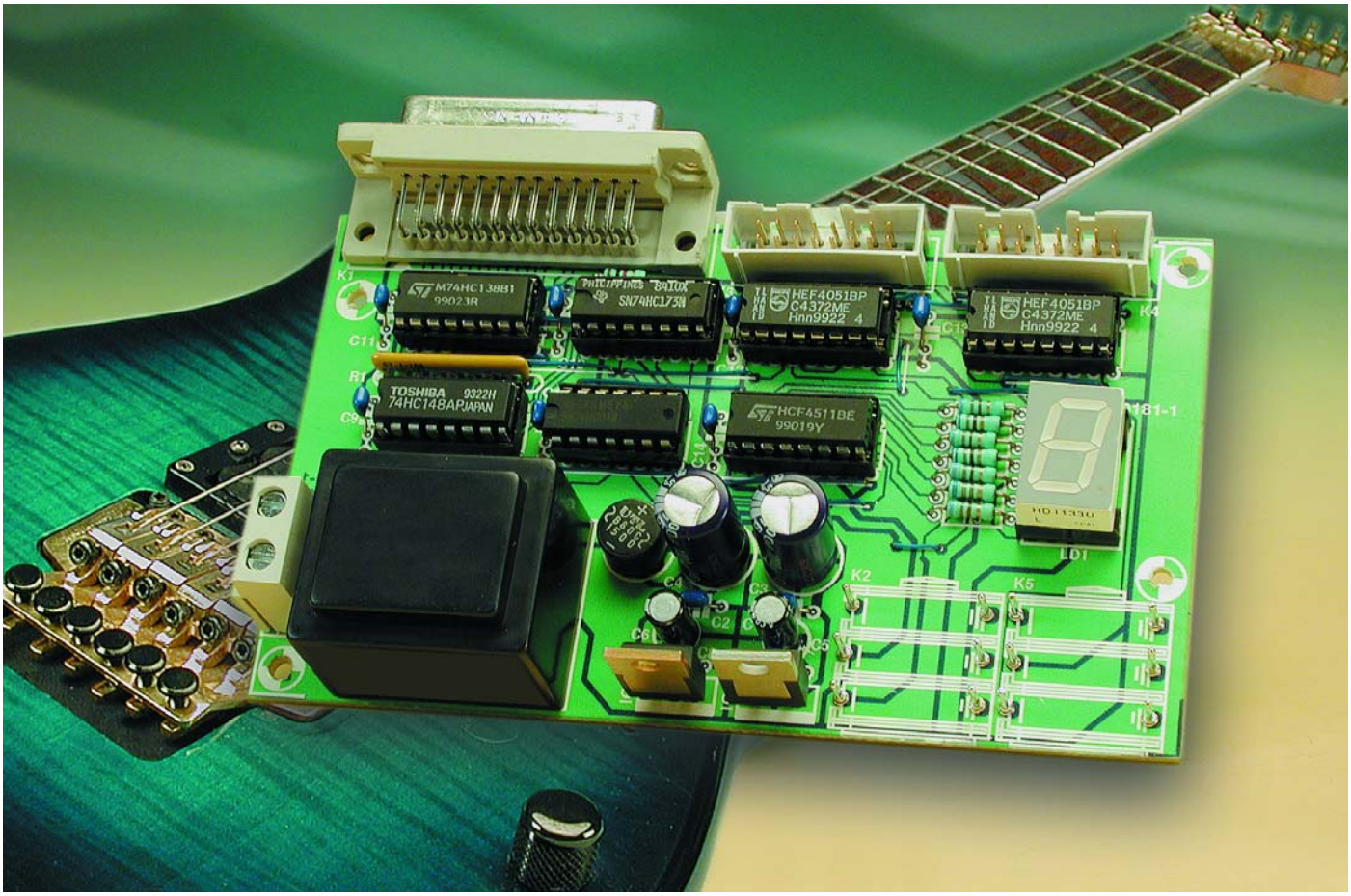
Semiconductors:
D1-D24 = bicolour LED, 2 pins, 3mm dia.
IC1-IC6 = 74HC164

Miscellaneous:
K1 = 5-way SIL pinheader
PCB, order code **010134-2**

Guitar Effects Switchbox

Design by T. Rooney Poms

This effects switchbox will help make the stage a safer and tidier place. All your effects units can be kept out of harm's way and seamlessly operated using a single footswitch.



Guitar effects units such as distortion, flanger or phaser are fiddly to switch on and off. Each effects unit has its own mechanical switch. In order to select a particular effect, first that effect must be switched on and then the others switched off (or vice versa). Since it is rather tricky to operate the switches simultaneously, a seamless changeover is impractical. The circuit described here solves this problem in a simple and effective way. It is not limited to switching guitar effects in and out, but is suitable for similar applications where analogue signals are passed through a chain of units.

The guitar effects switch is foot-operated. As usual for such devices, the control for the unit is a separate footswitch, the remainder of the circuit (i.e. the logic circuitry for the effects switch and the effects units

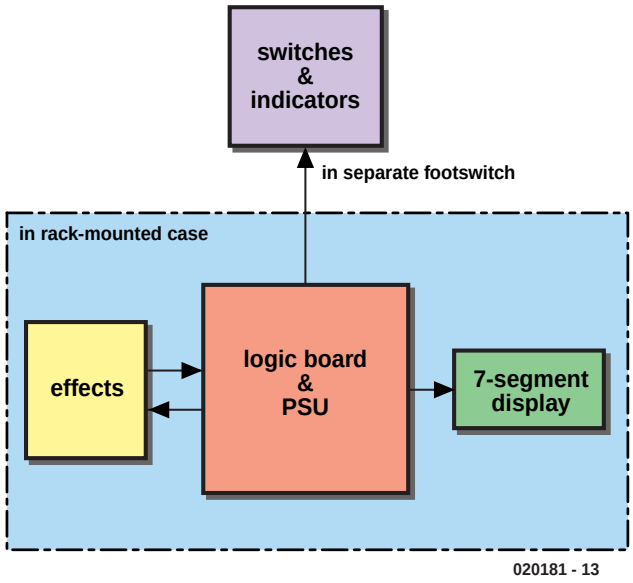


Figure 1. The effects switch and the effects units are kept in the rack: only the footswitch appears on the stage.

themselves) being mounted in a rack (**Figure 1**). The trick is that the effects are not selected using switches, but rather using buttons. The logic ensures that the effect remains active when the foot is removed from the button until the next time a button is pressed. This means that cables carrying audio signals are kept safely in the rack rather than being trailed around the stage. The connection between the foot control and the logic circuit carries just the switching signals.

Keyboard encoder

At the heart of the circuit in **Figure 2** is an 8-input priority encoder type 74HC148, with D-type flip-flops (IC2) connected to its output. This device generates a 3-bit control signal according to the button that has been pressed. The inputs to the 74HC148 are active low; when no button is pressed, all inputs are high. When a button is pressed the corresponding input is taken to ground. A binary encoded signal then appears at outputs 1A to 3A, which is stored in three flip-flops of the 74HC173. In order that this value is held when the button is released, the 74HC173 must be clocked to latch the encoder's outputs. This clock can be derived from the GS (A) output (pin 14) of the priority encoder. The output swings from high to low when a button is pressed. The signal must therefore be inverted (using a 74HC04 inverter) to clock the 74HC173. Not only that: in order to ensure that the IC is only clocked when the 3-bit control signal is valid, two further inverters provide an small additional delay to the clock signal. The three remaining inverters are not used and so have their inputs connected to ground. The 3-bit control signal from the output of IC2 forms a bus that is connected to four ICs.

Analogue switches and displays on the bus

Two of these ICs are type 74HC4051 analogue multiplexers. Up to eight effects units can be connected between these multiplexers via connectors K3 (input) and K4 (output). The guitar is connected to connector K2, the amplifier to K5. The 4051 is a low-cost choice for the switch: a pin-compatible alternative is the MAX4617 from Maxim which, although much more expensive and harder to obtain, has superior performance: it offers a very low R_{ON} and an excellent channel-to-channel separation of –93 dB. A comparison of the data sheets shows the clear difference in performance:

http://pdfserv.maxim-ic.com/arpdf/MAX4617-MAX4619.pdf

http://www.philipslogic.com/products/hc/pdf/74hc4051.pdf

The two other ICs connected to the 3-bit control bus drive displays to shown the selected channel or effect. The channel number is shown on a common cathode 7-segment display on the device itself, which is driven by BCD-to-7-segment decoder IC6 (type 4511). The unused control input of the 4511 (pin 6) is connected to ground. The 4511 is capable of driving 10 mA per output, and this value should not be exceeded. The value of the series resistors required for the 7-segment display will depend on the type chosen. For the type recommended in the parts list, 680 Ω is suitable.

The selected channel is also indicated by LEDs on the foot control. For this IC3, a 74HC238 3-to-8 line decoder/demultiplexer, is used to convert the 3-bit control signal into individual outputs. The eight outputs of the IC are connected to LEDs D1 to D8 in the foot control via a 25-way D connector. Since the 74HC238 does not have a particularly high output drive, low-current LEDs are required in the foot control, with a forward current of no more than a few milliamps.

All the ICs connected to the 3-bit control bus operate 'transparently'; that is to say, that they do not require clock or enable signals to be activated before processing their inputs.

A regulated 5 V supply is required for the logic circuitry, while the analogue part (the two analogue multiplexers) requires a symmetrical supply to pass the audio signals through faithfully. For this reason a symmetrical power supply with a 9 V mains transformer with two secondary windings is required, along with two voltage regulators (a 7805 and a 7905). Since the maximum current consumption of the circuit is only around 10 mA, heatsinks are not required for the voltage regulators.

Construction

Now we look at the construction of the circuit. As will be seen from the printed circuit board layout in **Figure 3**, the layout has not been specially designed for use in a rack enclosure. This makes for a compact and economical single-sided printed circuit board at the cost of a certain amount of extra wiring. First fit the several (insulated) wire links near to and under the ICs, and then proceed with

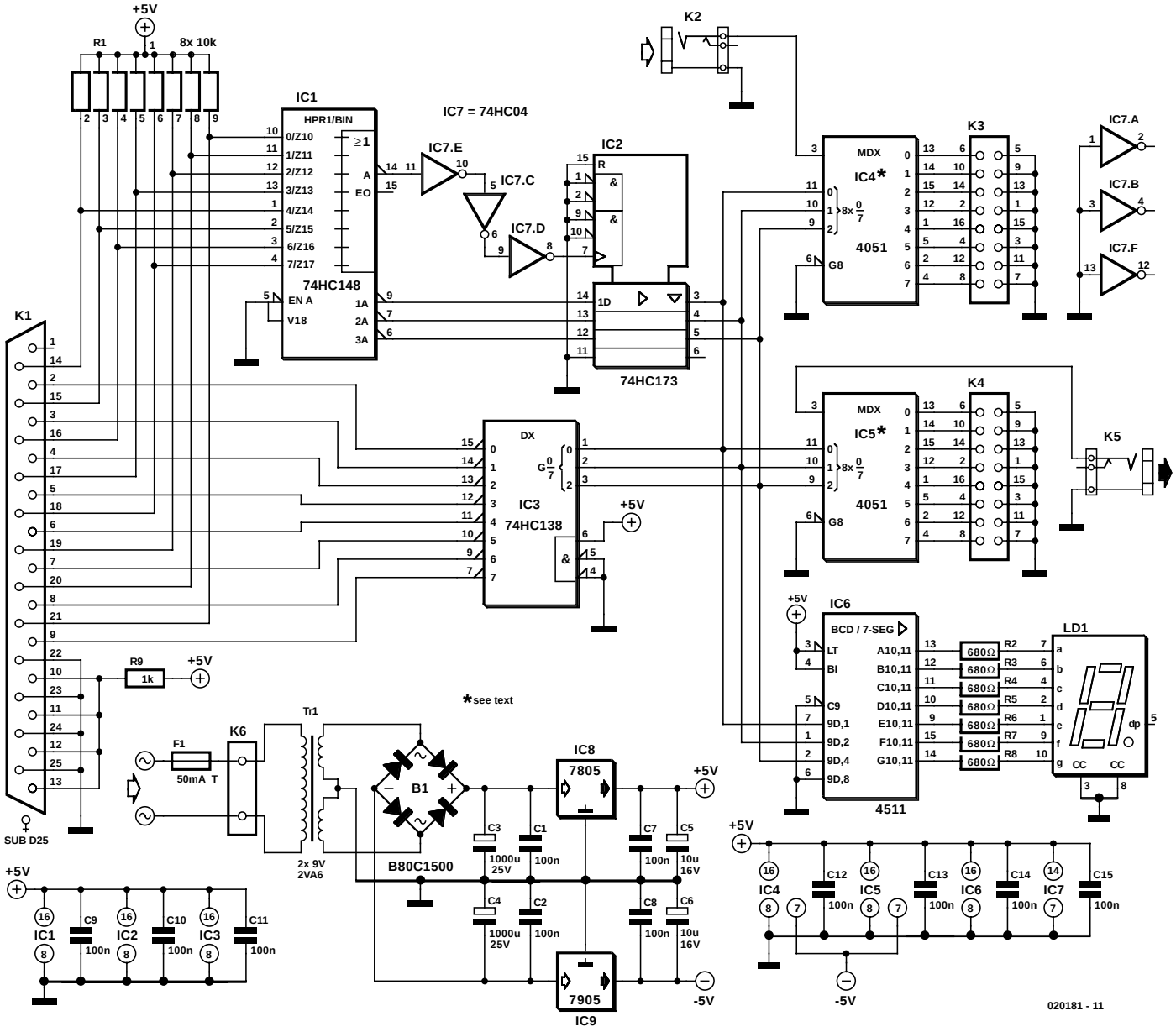
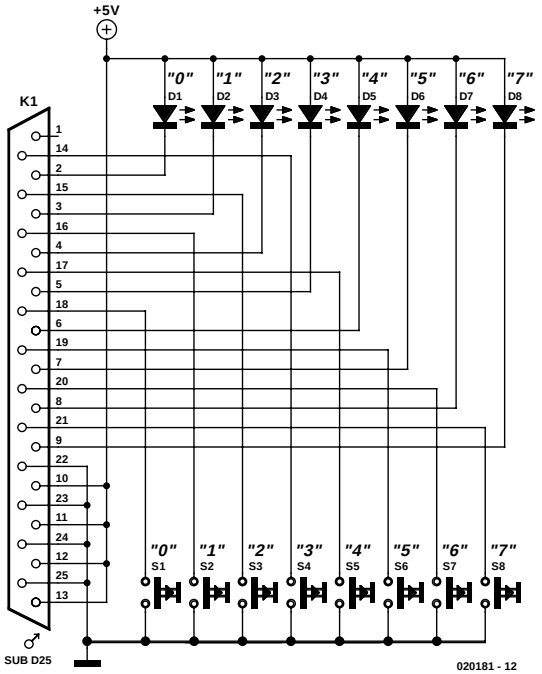


Figure 2. The effects switch circuit employs only standard logic ICs.



the resistors, capacitors and IC sockets (or preferably good quality SIL socket strips). Finally fit the larger components: the jack sockets, transformer and connectors. One point to note: the 7-segment display can easily be mounted directly on the front panel next to the two jack sockets by using a right-angled IC socket (for example Conrad Electronics # 188379).

In order to test the circuit before embarking on a large amount of cabling, it is best to construct the footswitch first. As far as we know, enclosures for eight footswitches are not available: the biggest we could find was a six-way unit from Marshall, and so home construction is

COMPONENTS LIST

Effects switch

Resistors:
R1 = 8-way SIL array, 10kΩ
R2-R8 = 680Ω

Capacitors:
C1,C2,C7-C15 = 100nF radial
C3,C4 = 1000µF 25V radial
C5,C6 = 10µF 16V radial

Semiconductors:
B1 = B80C1500 bridge rectifier in round case (80V piv, 1.5A)
IC1 = 74HC148
IC2 = 74HC173
IC3 = 74HC138
IC4,IC5 = 4051 or MAX4617*
IC6 = 74HC04
IC8 = 7805
IC9 = 7905
LD1 = 7-segment display with common cathode (e.g., HD1133O, Siemens)

Miscellaneous:
K1 = 25-way sub-D socket (female), PCB mount, angled pins
K2,K5 = mono jack socket, 6.35 mm, PCB mount
K3,K4 = 16-way boxheader
K6 = 2-way PCB terminal block, lead pitch 7.5mm
Tr1 = mains transformer 2×9V/2.6VA (Hahn BV EI 303 2016)
IEC mains appliance inlet with integrated fuse holder
16 mono jack sockets, 6.35mm, chassis mounting
PCB, order code **020181-1** (see Readers Services page)

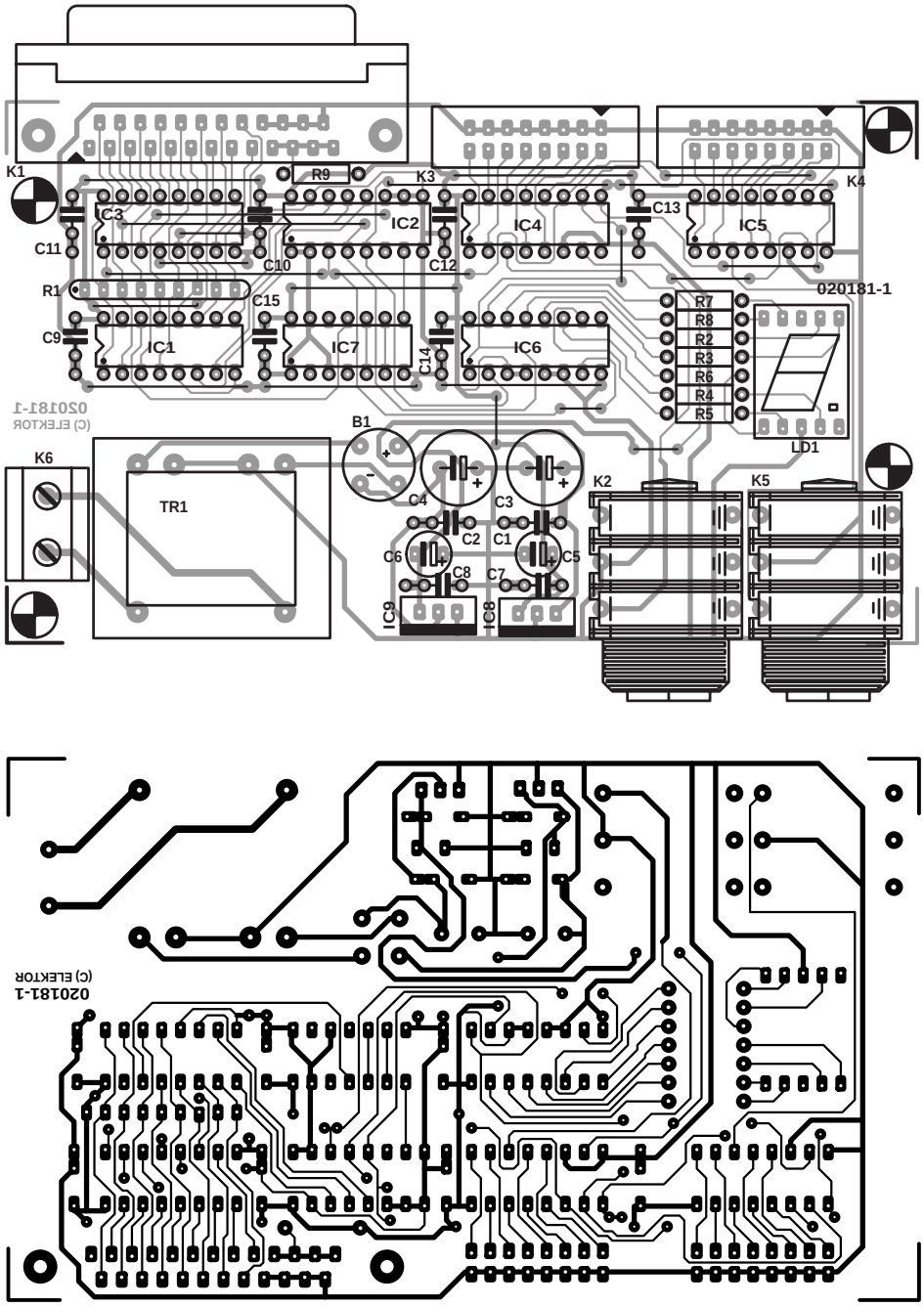


Figure 3. The compact printed circuit board layout.

COMPONENTS LIST

Footswitch

Semiconductors:
D1-D8 = LED, 3mm, low current

Miscellaneous:
K1 = 25-way sub-D plug (male)
S1-S8 = footswitch (e.g., Monacor/Monarch FS-10)

required. This at least has the advantage that the dimensions of the unit can take the shoe size of the guitarist into account! A printed circuit board layout is not necessary, since the eight buttons, the eight LEDs and the 25-way D-type connector can be wired individually.

Once the footswitch is ready and connected to the control logic using an ordinary computer cable, the circuit can be tested. First, with the ICs not fitted, apply mains power and check that the correct voltages are found at the right pins of all the IC sockets. If so, switch off the power and fit the logic ICs. The circuit can now be quickly tested by pressing the footswitches in turn. If the appropriate number appears on the display and the correct LED lights on the footswitch, the circuit is working correctly. The two box headers can now be wired to the 16 corresponding jack sockets, using unscreened cable as long as the distance involved is no more than an inch or two.

(020181-1)

Practical Neural Networks (3)

COURSE SUPPORT WEBSITE
http://www.rgu.ac.uk/eng/compint/page.cfm?pge=6732

Part 3 – Feedback Nets and Competitive Nets

By Chris MacLeod and Grant Maxwell

This month we look at two more advanced neural nets. The Hopfield network, which uses feedback in its structure, and the Competitive net which can recognise patterns in data, even if the programmer doesn't know they exist.

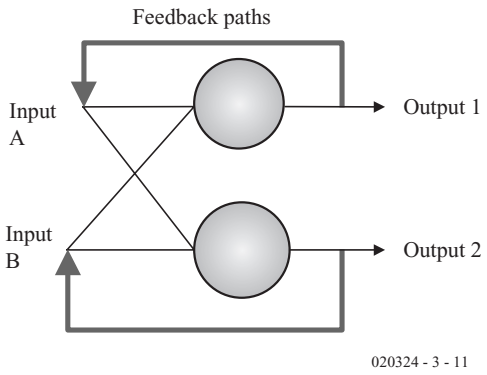


Figure 1. A Hopfield Net.

In 1983 a physicist named John Hopfield published a famous paper on neural nets. This paper helped to re-ignite the field, which had been languishing in the doldrums for some time.

Actually, the ANN which bears his name — the Hopfield Network — is of somewhat limited practical use, but it does help us to understand the ins and outs of neural net behaviour.

What Hopfield did was to create a network with **feedback** — with connections from the outputs, back towards the inputs. **Figure 1** shows the idea. The network always has a single layer and the same number of inputs as neurons.

Forward pass operation

The neurons operate in the same way as the binary ones described in part 1 (except that they produce a –1 and +1 output, rather than 0 and 1). The only difference in operation is that the output of the network is fed back to the input once it's been calculated and so goes through the network again. Eventually, if the network has been trained properly, the output will become constant (the inputs and outputs will be the same). The network is said to have **relaxed**. The network is then ready

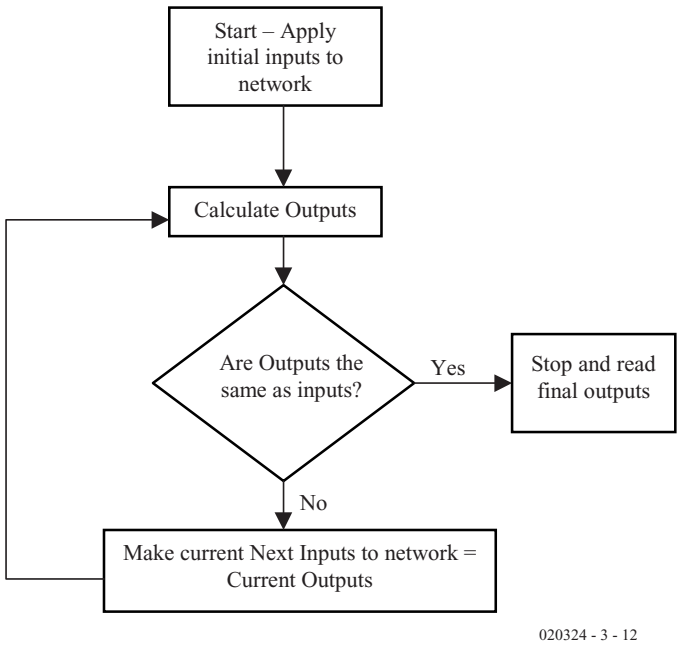


Figure 2. Running data through a Hopfield net.

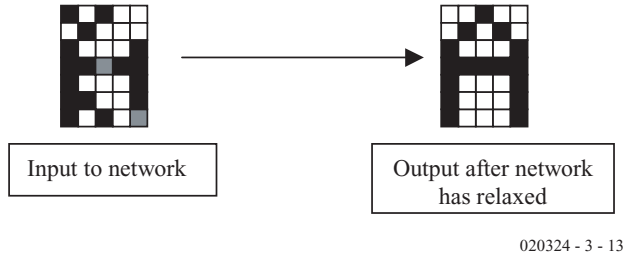


Figure 3. Operation of a Hopfield net.

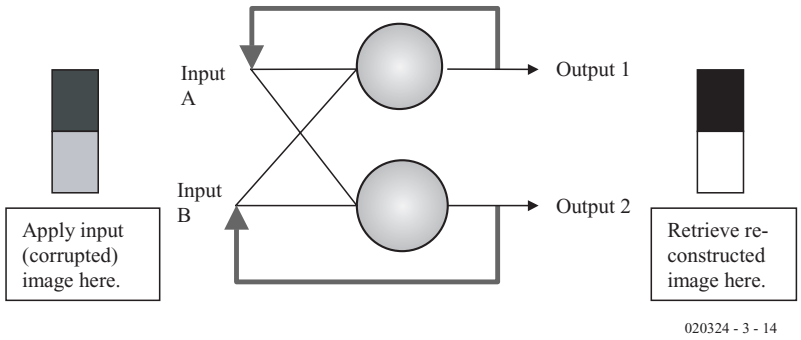


Figure 4. Applying images.

for you to read its outputs. **Figure 2** shows this process in the form of a flow chart.

Uses

Before going any further, it's worth pausing to consider what it is that the Hopfield network can do, which the BP network can not. A Hopfield network, rather than just recognising an image, can store and retrieve patterns — it has a memory. We can input a corrupted image into the network and it will reproduce the perfect stored version. **Figure 3** shows the idea.

Once the network is trained properly, all we have to do is present it with the corrupted version as its inputs and then wait until the network stops cycling as described above. Once this has happened, we can read the outputs of the network and they will give us a reconstructed image, see **Figure 4**.

In the original Hopfield net, all inputs and outputs are –1, which could represent, say, a white pixel and +1 for a black pixel. Networks with continuous outputs are today more common, but for our discussion, we'll stick with the simple case.

Training

Now that we know what the Hopfield network does, let us turn our attention to how it can be trained.

Compared with the Back Propagation network, training the Hopfield is easy. All the weights are calculated using a simple formula:

$W_{m,n} = \Sigma\, O_m\, O_n$ Over all patterns. Make weights $W_{n,n} = 0$.

Where $W_{m,n}$ is the weight of the connection between the m_{th} input and the n_{th} neuron and O_n is the n_{th} output desired from the network.

In other words, to find the weight of the connection between input m and neuron n, take each pattern to be trained in turn and multiply the m_{th} output by the n_{th} output and add them all together. As usual this is best illustrated by example, see **Figure 5**.

Let's say we'd like to train three patterns:

Pattern number one:

$O_{A(1)} = -1\ O_{B(1)} = -1\ O_{C(1)} = 1$

Pattern number two:

$O_{A(2)} = 1\ O_{B(2)} = -1\ O_{C(2)} = -1$

Pattern number three:

$OA(3) = -1\ OB(3) = 1\ OC(3) = 1$

$w_{1,1} = 0$

$w_{1,2} = O_{A(1)} \times O_{B(1)} + O_{A(2)} \times O_{B(2)} + O_{A(3)} \times O_{B(3)} = (-1) \times (-1) + 1 \times (-1) + -1 \times 1 = -1$

$w_{1,3} = O_{A(1)} \times O_{C(1)} + O_{A(2)} \times O_{C(2)} + O_{A(3)} \times O_{C(3)} = (-1) \times 1 + 1 \times (-1) + (-1) \times 1 = -3$

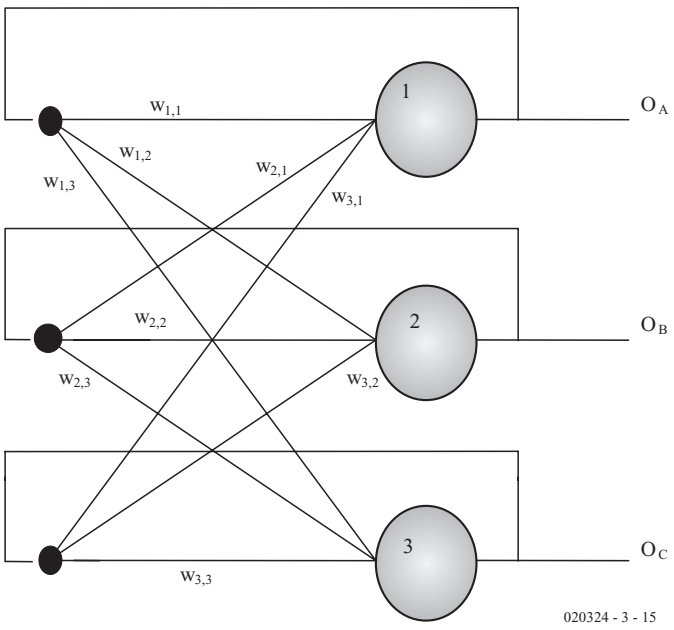


Figure 5. Worked example of Hopfield training.

$w_{2,2} = 0$

$w_{2,1} = O_{b(1)} \times O_{a(1)} + O_{b(2)} \times O_{a(2)} + O_{b(3)} \times O_{a(3)} = (-1) \times (-1) + (-1) \times 1 + 1 \times (-1) = -1$

$w_{2,3} = O_{b(1)} \times O_{c(1)} + O_{b(2)} \times O_{c(2)} + O_{b(3)} \times O_{c(3)} = (-1) \times 1 + -1) \times (-1) + 1 \times 1 = 1$

$w_{3,3} = 0$

$w_{3,1} = O_{c(1)} \times O_{a(1)} + O_{c(2)} \times O_{a(2)} + O_{c(3)} \times O_{a(3)} = 1 \times (-1) + (-1) \times 1 + 1 \times (-1) = -3$

$w_{3,2} = O_{c(1)} \times O_{b(1)} + O_{c(2)} \times O_{b(2)} + O_{c(3)} \times O_{b(3)} = 1 \times (-1) + (-1) \times (-1) + 1 \times 1 = 1$

Unlike BP training, the calculations are done only once and are not repeated.

We can write a simple algorithm to set the weights for a Hopfield as shown in **Listing 1**.

Where the same variables are used as shown in the forward pass example in part 1 (where the weights are held in a two-dimensional array). The desired outputs are held in an array i(pattern_no, pixel_number).

Capabilities

So the Hopfield net has a memory. But what else can it do? Actually, its practical applications are a little limited, but it tells us a lot about the general capabilities of neural nets.

In part 1, we discussed the similarity of the feed forward network to combinational logic. But the ANN is logic which can produce any truth table by learning, rather than detailed design. Similarly, the analogy for the Hopfield is sequential logic. After all, a flip/flop like a JK or SR is a simple memory and this is also achieved through the use of feedback.

In fact, the Hopfield can produce time-series, oscillatory or even chaotic outputs if you let it; although the training illustrated above is designed always to produce a stable network — the outputs always decay to a steady state.

The simple Hopfield net illustrated here has limitations. It is prone to local minima (which in this case means that it may have difficulty reconstructing some patterns), so more sophisticated training algorithms have been developed. For example, there are variants of the Back Propagation algorithm which can be used to train Hopfield nets using targets (like the BP networks in part 2), but these don't guarantee stability.

We can extend the capabilities of the simple Hopfield if we add an extra layer. Such networks are known as **Bi-directional Associative Memories** (BAMs) and can associate an input with a different memory. But beyond this, the structure of the Hopfield net is too rigid, we need to use its lessons to devise more general nets.

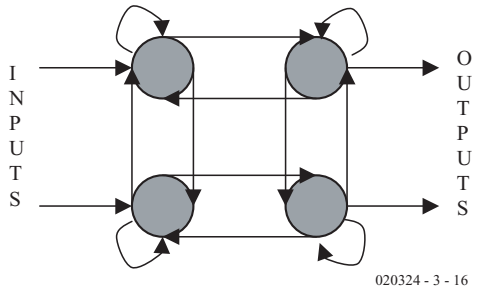


Figure 6. A general neural net.

Listing 1

```
FOR f = 1 TO no_of_inputs
  FOR t = no_of_inputs + 1 TO no_of_inputs + no_of_outputs
    FOR p = 1 TO no_of_patterns
      w(f, t) = w(f, t) + i(p, f) * i(p, t - no_of_inputs)
    NEXT p
    IF t = no_of_inputs + f THEN w(f, t) = 0
  NEXT t
NEXT f
```

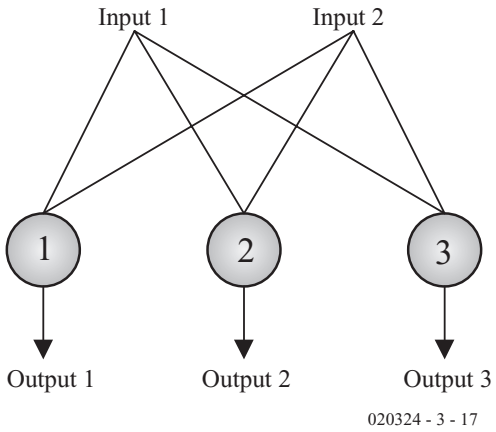


Figure 7. A simple competitive net.

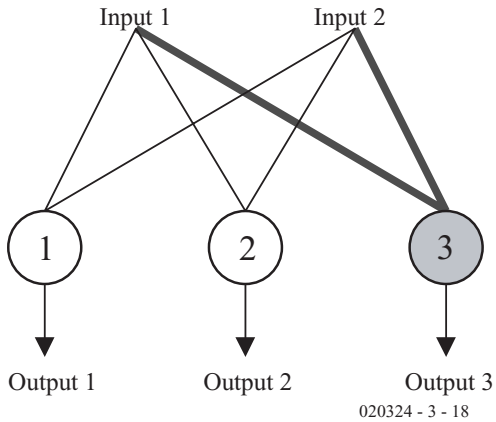


Figure 8. A winning neuron.

General Neural Nets

We've seen how the Hopfield net is more general than the simple feedforward type. In fact the feedback type just degenerates to a feedforward net if the feedback paths are set to zero.

You might guess therefore, that the most general neural nets would have a mixture of both feedback and feedforward connections. In fact this is true. In the most general network, any neuron could be connected to any other, **Figure 6** shows the idea.

Training such networks is tricky, as algorithms like the Hopfield Training illustrated above and even Back Propagation only operate when the network has a defined and limited structure. To train a network where any neuron may be connected to any other demands more advanced algorithms.

Perhaps the easiest algorithm to employ and certainly the most common in such circumstances is the **Genetic Algorithm**. One can employ the algorithm to choose the weights in a general network in the same way as one can use it to choose component values in the examples given in that article, the fitness of the network being the inverse of its error. The details of such advanced training methods can wait for a future article.

Competitive Learning

Now, let's look at a quite different network. You'll remember that in part 2, we mentioned that probably 80% of neural nets used in practice were Feedforward, Back Propagation nets. This leaves the question of the remaining 20%. Well, most of these are of a type of network known as a **Competitive** or **Winner-Takes-All** Net.

Operation

The **Competitive** net is best illustrated by example. Suppose we have a network of three neurons as shown in **Figure 7**.

The neurons work in exactly the same way as those already described in part 1, except that we don't need to apply a threshold or sigmoid function.

We won't worry too much at this stage about the set up of the weights except to say that they are essentially random.

Now let us apply a pattern to the network. Just by chance (since the weights are random), one of these neurons will have a higher output than the others — let's say it's neuron three, as shown in **Figure 8**.

We say that this neuron has **won** and set its output to 1 and the others to zero.

Now we train **only** the weights of neuron 3 (the ones shown in bold), so that, if this pattern comes along again it will have an even higher output — it will win even more easily. So neuron three will always fire when this pattern comes along. In other words, neuron three recognises the pattern. This is very simple to do; we just update the weights with this formula:

$$W^{+} = W + \eta(\text{Input} - W)$$

Where W^{+} is the new (trained weight) and W is the original weight, Input is the input feeding that weight and η is a small constant, much less than 1 (say 0.1).

Of course if another, completely different, pattern comes along a different neuron will win and then this new neuron will get trained for that pattern and so the network **self organises** so that each neuron fires for its own pattern.

Uses

Suppose we let a competitive network loose on some data — let's say from the financial markets. The network would organise itself to find patterns in the data. Exactly what these patterns are, we don't know, the network decides for itself — we don't give it examples like a Back Propagation network. This is both the attraction and the disadvantage of the Competitive network — you might find an important pattern in the data which you didn't know existed — but it could miss what you're looking for and find some other unwanted patterns to recognise.

In the same way and related to this, the network will fire the same neuron for patterns **it** finds similar

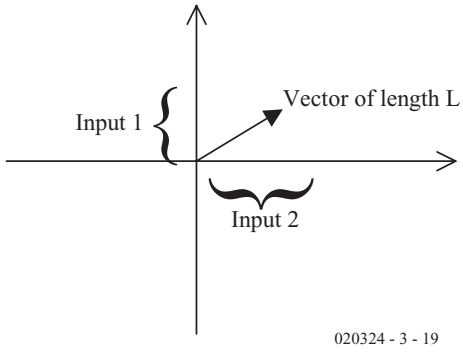


Figure 9. The inputs shown on a graph.

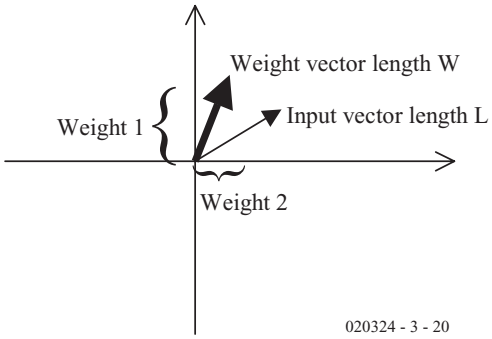


Figure 10. The weight vector of neuron 3.

(even although the similarity may not be obvious to the user). We can say therefore that, whereas the Back Propagation network is trained by the user to recognise patterns, the Competitive net trains itself to classify patterns.

Of course you could use the competitive neuron to recognise patterns like Back Propagation. But this seems rather a waste of effort since BP works extremely well and is generally easier to set up than a competitive net.

More detail

To understand some of the subtle features of the competitive system, we need to examine its operation a little more closely. To do this, let's look at the network shown in Figures 7 and 8 more closely.

The network has two inputs and it's possible to represent these as a line (called a vector) on a graph, where y is the value of input 1 and x input 2. This is shown in **Figure 9** (of course this applies to any number of inputs, but two are easy to visualise).

The length of this vector by Pythagoras is:

$$Length = \sqrt{(input1)^2 + (input2)^2}$$

We can also plot a line representing the weights of neuron 3 on the same graph by making its two weights the x and y coordinates, as shown in **Figure 10**.

Now, when we work out the output of the neuron ($i_1w_1 + i_2w_2$), what we are actually doing is calculating what's known as the **dot product** — which can be considered a measure of how similar the two vectors are. If both the vectors were exactly the same (one lying on top of the other) the output would be larger than if they were different.

If all the vectors were the same length, then we'd just be measuring the angle between them (which would make matters easier, as it means that we don't have to take length into consideration), so that is what we do. We can make all the vectors one unit long by dividing them by their length.

Now, consider the weight vectors for all three neurons in the network, **Figure 11**. These have all been normalised to one unit as described above.

Neuron 3 has won because it is closest to the input and therefore has the largest dot product (it is most similar to the input). What the training does, is move the weight vector of neuron 3 even closer to the input, as shown in **Figure 12** (remember that only the weights of neuron 3 are trained).

This, of course, makes it likely that, if a similar pattern comes around again, neuron 3 will fire.

The training formula $W^+ = W + \eta(\text{Input} - W)$ doesn't preserve the unit length of the weight vector it's operated upon; so after using it, you should divide the weight vector of the winning neuron by its length to make it one unit again.

You can probably see that the distribution of the weights in this type of network is quite critical and so it helps to consider the distribution of vectors around the origin when setting up the network to ensure an even coverage.

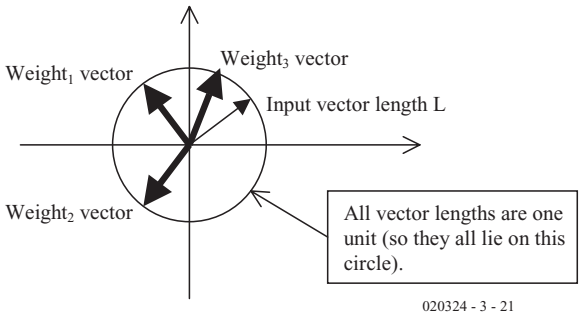


Figure 11. Weight vectors for all three neurons.

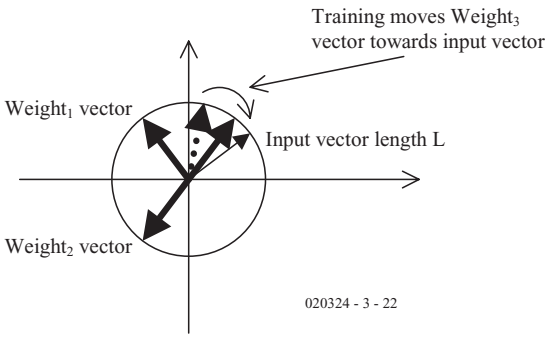


Figure 12. Effect of training.

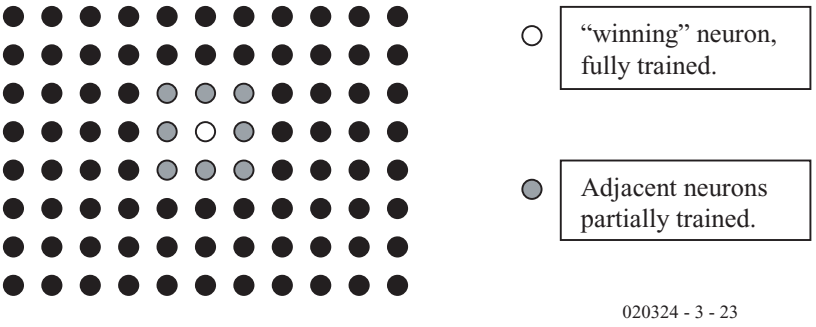


Figure 13. A Self Organising Map.

Networks based on Competitive Neurons

Competitive neurons are seldom used just on there own, but form the mainstay of several more complex networks. They are often laid out in the form of a 2D grid as shown in **Figure 13**. This is known as a **Kohonen Self Organising Map**.

What happens in this case is that the winning neuron (shown in black) is fully trained and the surrounding neurons (shown in grey) are partially trained (by making η in the formula, a smaller number).

When the network has been allowed to train in this way the result is that it forms a map in which most similar patterns are grouped together and are far away from the less similar ones.

Another very advanced network based on Competitive neurons is **Adaptive Resonance Theory** (ART). This network can change its size and grow as it learns more patterns.

In the final part of the series we'll have a look at some of the other applications of neural nets and some of the more advanced topics which researchers are wrestling with.

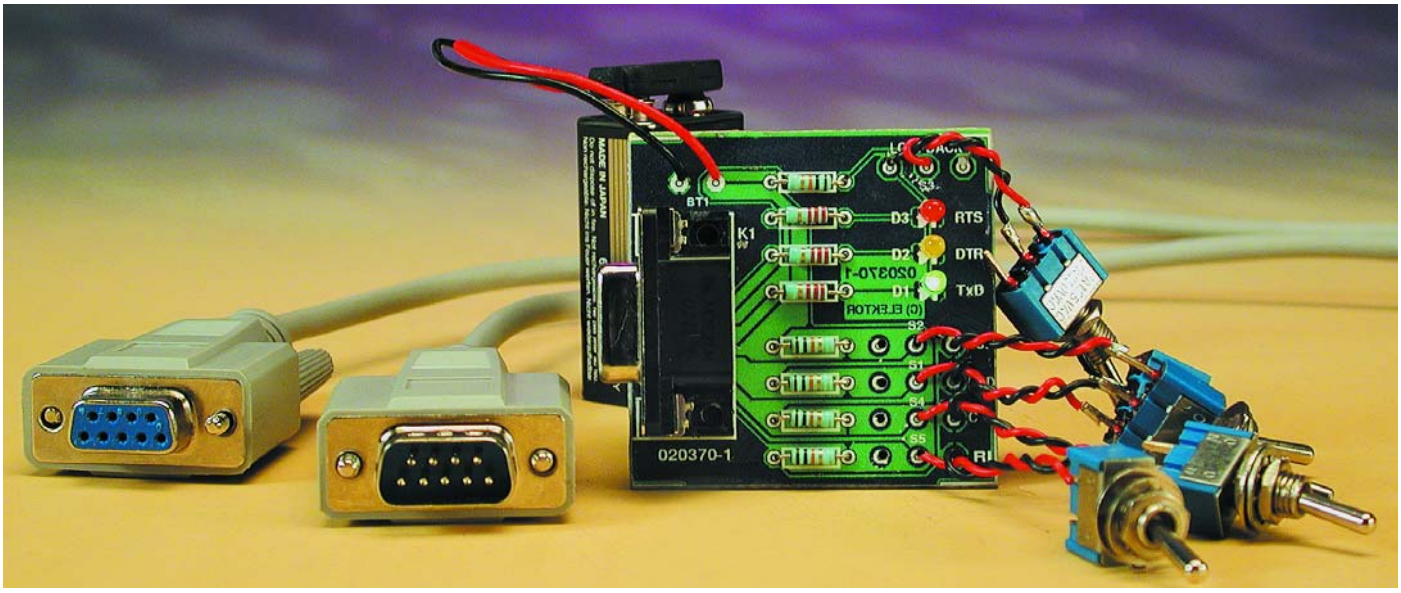
(020324-3)

COM Port Tester

With 3 LEDs and 4 switches

Design by P. Goossens

The serial port (often called the COM port) is perfect for connecting various electronic circuits to the PC. The complexity of the connected circuit varies from a few I/O lines to control something and read the state of switches, to a complex communications system between several computers. This circuit is in the first category and uses the serial port to perform a few simple I/O tasks.



The COM Port Tester is a simple circuit that shows how the signals on a serial port can be used to control another electronic circuit. We will have to disappoint those of you who expect to use the serial port to control motors and lights etc. This circuit is limited to driving three LEDs and reading the state of four switches, nothing more and nothing less.

RS-232

The functions and electrical characteristics of the signals in a serial port have been laid down in a standard. This standard is called the RS-232 standard and was drawn up early in 1960 by the 'Electronics Industries Association'. The aim was to define a standard port that could be used with equipment from various manufacturers to communicate between each other. A good 40 years later this standard (although with a few modifications) is still in widespread use in all kinds of equipment. The best known of these are of course the mouse, modem, etc., but in industrial equipment the presence of an RS-232 port is also common.

In the RS-232 standard equipment has been divided into two distinct groups, these being DTE and DCE. These abbreviations stand for 'Data Terminal Equipment' and 'Data Communication Equipment' respectively. These two terms were chosen by the authors of the standard due to the way in which the serial port was used at that time: on one side of the connection was a terminal, which was nothing more than a screen with a keyboard, and on the other end of the link was some sort of

communication equipment.

The function of a terminal (DTE) is very simple in this set-up: every key that is pressed is transmitted via the serial link, and every character received is displayed on the screen. The communication equipment (DCE) takes care of all other aspects of the communication. The serial port is obviously not limited to the connection of terminals to communication equipment. Even so, DTE and DCE are still used to describe the two distinct groups of equipment.

Signals

The RS-232 standard describes many more signals than are required on the PC's serial port nowadays. This made it possible to use a 9-way connector instead of the original 25-way connector. The extra signals that were found on the 25-way connector are (almost) never used in modern equipment. Those signals that are in use can be found in **Table 1**. The direction of the signals is also included. 'DCE to DTE' means that the signal originates from the DCE (output) and ends up at the DTE (input). 'DTE to DCE' obviously means the opposite. The two most important signals are RxD and TxD. These signals are used for the exchange of the serial information ('serial' meaning one after the other, and in the case of the RS-232 port it is bit by bit).

A DTE can be easily distinguished from a DCE by looking at the connector used. If the equipment has a male connector for the serial port, then it is a DTE (for example a PC). If instead the equipment has a female connector, then the equipment concerned is a DCE (for example a modem, home-built circuit, etc.).

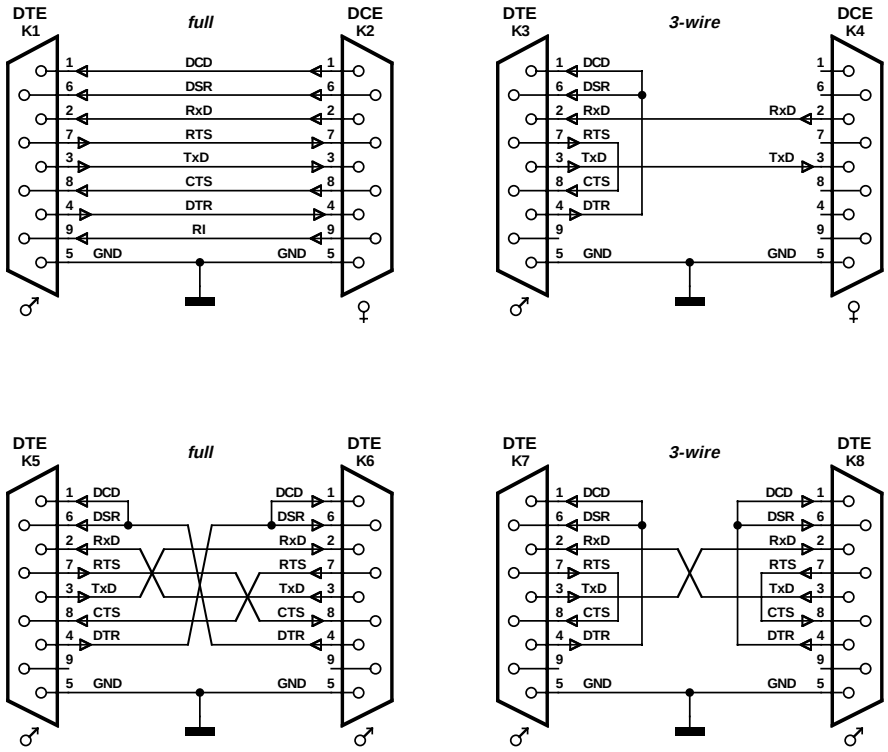


Figure 1. Several types of cable can be used to make a serial connection.

Electrical aspect

The signals on the RS-232 port obviously have to conform to an electrical specification. This is where the authors of the standard have decided on something that may appear peculiar to the digitally minded amongst you. A logical one (called a MARK in serial communications) is electrically represented by a signal between –3 V and –25 V, whereas a logical zero (RS-232 term: SPACE) has a voltage level between +3 V and +25 V. So not only are the voltages different compared to TTL levels, but all signals are also inverted.

Most PCs are now quite happy with TTL level signals, where a voltage between 0 V and 0.8 V corresponds to a '1', and a voltage between 2 V to 5 V corresponds to a '0'. Of course the modern RS-232 ports can still withstand voltages between –25 V and +25 V, so that equipment that conforms to the RS-232 specification won't cause any damage!

Cabling

Choosing the cable required to connect two pieces of equipment together via the serial port often causes confusion due to the many possible permutations. There is a big difference between connecting two DTEs (two PCs for example) and connecting a DCE to a DTE (a modem to a PC for example). Some of the different configurations can be seen in **Figure 1**.

When you examine these cables it is noticeable that for connections between a DTE and DCE no signals cross over, whereas for connections between two DTEs some signals do cross over. The number of signals that is required to make a connection between two pieces of equipment can vary from three to nine. The 25-way version has even more signals that can be connected. These signals have been ignored here for the sake of simplicity. They are also no longer very important, since 25-way connectors are hardly ever

Table 1. The signals on a 9-way connector.

DCD (Data Carrier Detect)	DCE to DTE
RxD (Receive Data)	DCE to DTE
TxD (Transmit Data)	DTE to DCE
DTR (Data Terminal Ready)	DTE to DCE
DSR (Data Set Ready)	DCE to DTE
RTS (Request To Send)	DTE to DCE
CTS (Clear To Send)	DCE to DTE
RI (Ring Indicator)	DCE to DTE

used any more for serial connections.

Usually a 3-wire connection is sufficient to achieve communications, but some equipment requires that all signals are connected. These extra signals are used for hardware handshaking, which we won't cover in this article.

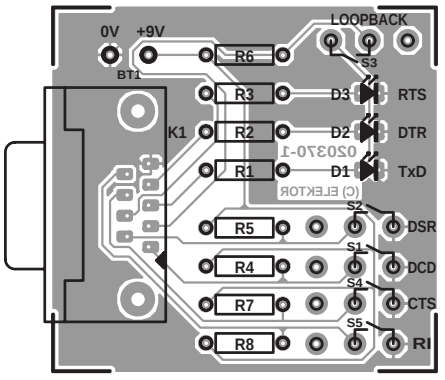
Circuit

The COM port tester (**Figure 2**) has been kept very simple. The main reason for this is that we did not keep to the exact RS-232 standard, which has a voltage between –3 V and –25 V for a logical one and a voltage between +3 V and +25 V for a logical zero. As was mentioned earlier, PCs are quite happy with TTL levels, so there is no longer a need to use a negative voltage to send a logical one to the PC.

All of the inputs and outputs found on a 9-way connector have been used in this circuit to communicate with the outside world. Here that was implemented using a few switches and LEDs.

An extra function is provided by switch S3. When this switch is closed, all serial data transmitted from the output of the serial port is fed back to the input of the serial port.

Figure 3 shows the layout of the PCB for this circuit. Considering the small number of components required, it shouldn't be a problem if you used a small prototyping board instead.



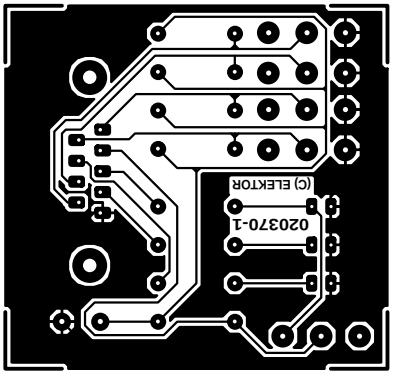


Figure 3. The PCB layout for the circuit in Figure 2 (board available from The PCBShop).

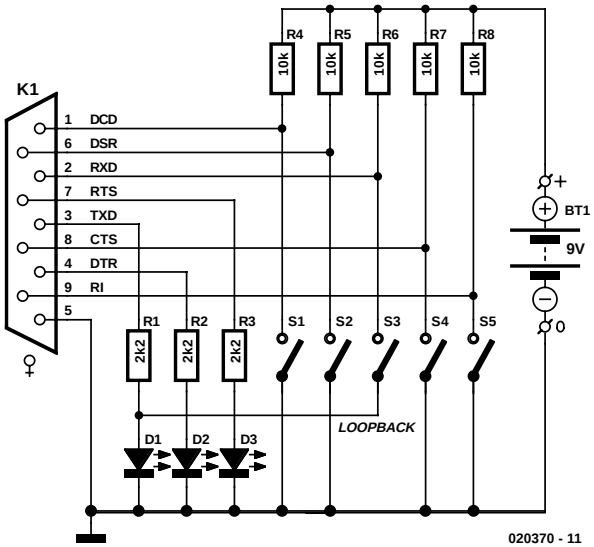


Figure 2. The circuit of the COM port tester consists of just a few resistors, switches and LEDs. A 9 V battery is used for power.

The circuit is connected to the PC via a one-to-one cable (the top-left cable shown in **Figure 1**).

Software

The original purpose of this circuit was to serve as an example to show how data could be processed in a simple manner via the serial port. Everybody will have their own ideas how this circuit can be put to use. This requires a program that has been adapted to the specific functions of the circuit.

The writing of such a program will be left as an exercise to the reader, but to start you off there is an article elsewhere in this issue of Elektor Electronics about a newly developed serial port driver for Windows. This is of course mainly of interest to people with some programming experience.

An example program for this article is obtainable from the publishers (order number **020388-11**, available on floppy disk or as a free download from the *Elektor Electronics* website). This permits the outputs of the serial port to be controlled manually, and also shows the state of the inputs.

After starting the program you can select the serial port required. Next, the port has to be opened by clicking on the button marked 'open'. The program is now ready for use and will show the state of the inputs using a tick in the boxes to the right of the connector.

When a box contains a tick it indicates that the relevant input is at a low voltage, which for an RS-232 signal means a logical one. This corresponds to a closed switch in our circuit.

COMPONENTS LIST

Resistors:
R1,R2,R3 = 2kΩ2
R4-R8 = 10kΩ

Semiconductors:
D1 = LED, 3mm, green, low-current
D2 = LED, 3mm, yellow, low-current
D3 = LED, 3mm, red, low-current

Miscellaneous:
K1 = 9-way sub-D socket (female), angled pins, PCB mount
S1-S5 = slide switch or rocker switch, 1 make contact
9-V battery with clip-on leads
PCB, available from **The PCBShop**
Disk, project software, order code **020388-11** (see article 'Serial Port Driver for Windows')

Naming of RS-232 signals in electronic circuits

Often the naming of RS-232 signals in electronic circuit diagrams is misused. A female connector is frequently used for the serial port in a circuit, which means that the equipment should be considered as a DCE. This implies amongst other things that the pin where the serial data comes out should really be called RxD, but is usually identified in the circuit diagram as TxD. The opposite applies to the pin that receives serial data; this is usually called RxD, whereas the correct RS-232 term is TxD.

The circuit diagrams in Elector Electronics always show the pin that transmits serial data as TxD and the receiving pin as RxD, irrespective of the type of connector used.

This circuit is an exception to this 'rule', since there is no logical reason to exchange signals such as TxD and RxD in this case.

The three buttons to the right of the outputs of the connector are used to set the outputs high or low. This is made visible on our circuit via the LEDs.

Conclusion

This circuit has been kept very simple. Because of this, it only has a limited functionality. To increase its usefulness you could for example add a buffer stage to the outputs, which could then drive a relay for switching lamps. The switches could be replaced by an input stage with an opto-isolator, or a mercury switch etc., thereby allowing the acceptance of different information by the serial port of the PC.

These improvements have not been included here, since we wanted to keep the circuit small and simple. But there is nothing that stops you from extending the circuit yourself, for example turning it into a computer controlled time switch, burglar alarm, and so on. The possibilities are only limited by your imagination (and your knowledge of electronics of course!).

(020370-1)

Serial Port Driver for Windows

Programming with Serial.DLL

By P. Goossens

In the April issue last year we published an article that showed how to program the serial port correctly under Windows. The methods described in that article appeared to be a bit too complex for many people. For this reason we wrote our own DLL, which can be used to simplify the programming of the serial port. This DLL will be described in this article.

The serial port is still used by many electronics hobbyists to connect their computer to the outside world. This makes perfect sense, since such an interface is found on every standard PC and it also provides a few input and output signals as well as the serial communication channel. The USB port has these capabilities as well, but this requires some extra circuitry and many hobbyists would prefer to avoid that. In this context it's better to keep quiet about ISA and PCI cards too. The serial port still remains the favourite solution when a simple connection is required between a PC and some electronics.

Programming

The way in which serial ports are programmed has changed tremendously over the years. Under DOS they could easily be controlled directly, without affecting the operating system. Under Windows this changed completely. Although the earlier versions of Windows still permitted the direct access of the hardware, programming the serial port became much more complex. The more recent versions completely refuse to let programs access the hardware directly. Windows now takes responsibility for this and likes to stay in control. It's only in this way that Windows can keep control of all hardware access.

The only (proper) method for controlling the serial port under Windows is via the Windows API. A number of functions for the serial port has been included in the API (these were described in the previously mentioned April 2002 article). There is still a lot of work involved to control a serial port using these functions. It would be much easier if the basic routines required for controlling the serial port were provided by a DLL, so the programmer has no need to concern himself with the communications between the program and Windows.

Serial.DLL

Many readers will probably appreciate that we have written a DLL that can be used to control a serial port in a simple manner. Apart from transmitting and receiving characters, it is also possible to request the state of the input pins and set the state of the outputs.

An extra feature included in the

Table 1.

The complete list of functions and routines in the DLL.

OpenCOM
CloseCOM
IsPortOpen
GetPortNr
COMPortExists
SendCharCOM
ReadCharCOM
SetTxD
ResetTxD
SetRTS
ResetRTS
SetDTR
ResetDTR
GetCTS
DetDCD
GetDSR
GetRI
GetHandle
CheckInputs
BaudRateSet
ParitySet
BitPerByteSet
StopBitsSet

DLL is an option for the it to keep an eye on the serial port and send a Windows message when its status changes. With this it becomes unnecessary to poll the serial port at regular intervals. It is only when there is a change in status that the program is interrupted, similar to the way in which Windows informs a program that a key has been pressed. Some of you may have difficulty in using this routine, but there is no need to panic. You don't have to use this routine to operate the serial port; it has only been included as an extra for the more advanced programmer.

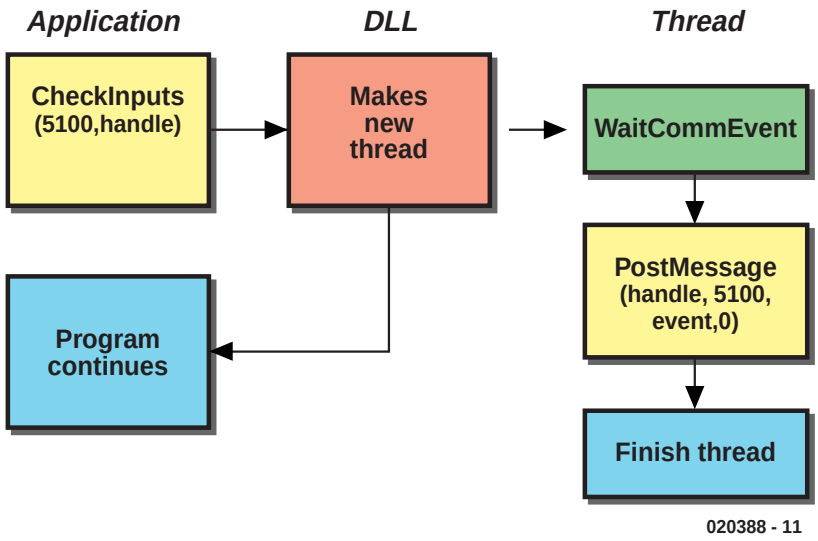


Figure 1. Implementation of the `CheckInputs` function.

The functions

The functions and routines of the DLL are listed in **Table 1**. All relevant functions have been named to make their purpose clear. We will therefore not go into detail for all functions. For this you should look at the accompanying example, which can be downloaded together with the DLL from our website at www.elektor-electronics.co.uk with the number **020388-11**.

Before a port can be used, it has to be opened. This is done by the `OpenCOM` function. This function requires the port number of the serial port that has to be opened. A 1 corresponds to COM1, a 2 is for COM2, etc. When the port has been opened successfully, the function returns the port number; when there has been a problem this value will be zero.

After use, for example when the program is terminated, the port has to be closed again. This is done by the `CloseCOM` routine. In between these events, characters can be transmitted using the `SendCharCOM` routine and received using `ReadCharCOM`.

The status of the inputs (DCD, DST, CTS and RI) can be obtained using the functions `GetDCD`, etc. The DTR output is controlled via the `SetDTR` and `ResetDTR` functions. Similar functions are provided for the two other outputs, TxD and RTS.

When the TxD output is driven directly you may run into problems. This output is normally used to transmit serial data. It's true that this output can be controlled directly, but if you attempt to transmit a character after the output has been set high with the `SetTxD` function, the transmission will fail and the program could crash. It is sufficient to reset the line using `ResetTxD` after which you can transmit as many characters as you like.

The above functions don't require much of an explanation. The opposite is true of the `CheckInputs` function. In order to use this function successfully you will first have to learn a bit about Windows messages. But don't let this put you off from using this DLL. The following section is only important if you want to make use of the facility to get a signal from Windows when the status of the serial port changes.

Windows messages

Programs can give commands to the operating system via its API, and in many cases Windows will issue a reply in one form or other. But this isn't the only way in which applications can communicate with Windows.

The messaging system is another important means of communication. Virtually every Windows program uses this. The operating system uses these 'Windows Messages' to let an application know when a mouse button has been pressed or if the program should be minimised, to give a few examples.

A message consists of no more than three variables that have to be interpreted by the application and subsequently acted on. The first variable specifies the type of message. The other two variables are used to pass on any other information that may be required. There wouldn't be much point if Windows told an application that the mouse pointer had moved, without telling it what the new position was.

There are numerous message types reserved by Windows, but it is also possible for a program to send a message to another application. For this Windows has reserved messages with a type value greater than 5000. These are the so-called User Defined messages and are therefore not sent by Windows, but can be can be used by applications for communicating between themselves. Most programming environments (such as Delphi, Visual C++, etc.) take care of these communications without requiring programmer involvement. But most programming environments also offer the programmer the facility to deal with these messages themselves when required.

With Delphi it is possible to use the `Application.OnMessage` event to process the incoming Windows messages. The corresponding routine in Visual C++ is `Application::OnMessage`. This communication mechanism is used by our DLL to inform the application whenever the serial port status changes. To send a message we can use the `PostMessage` function from the Windows API.

The Windows API also has a function that waits until there is a change in status of the serial port. This function is called `WaitCommEvent` and is suitable for our application, but has the problem that it halts the execu-

tion of the program until such time that there is a status change of the serial port. This is obviously not our intention. This problem can easily solved through the use of so-called threads.

Threads

An application can run parts of the program in a separate thread. This means that there are effectively two programs active at the same time. One is the main program and the other is the thread. If this thread is halted it won't affect the operation of the main program.

Figure 1 shows how the `CheckInputs` function is implemented in the DLL. The program calls this function, at which point the DLL creates a thread that is automatically started. From this point on the main programming continues to run normally, independently from the thread. The thread calls the Windows function `WaitCommEvent`, which causes Windows to halt the execution of the thread until there is a change in the status of the serial port.

Since Windows regards the thread as a separate program, the execution of our main program continues normally. In this case the program is running the `CheckInputs` routine, which is part of the DLL. The DLL completes that routine and the main program can continue with its tasks unhindered. As soon as the status of the serial port changes, Windows will re-start the thread and states in a variable which event occurred in the serial port. The thread subsequently passes this information on to our program via a Windows message.

When the program receives a Windows message it calls the function `AppMessage`. This function checks if the message indicates that the status of the serial port has changed (in this case we've used the value 5100, but this may be any other value as long as it hasn't already been used by Windows). If this is the case then a routine is called that deals with the status change. In all other cases the message has to be dealt with by the actual Message Loop.

And finally

We can imagine that a few question marks remain after reading the previous section. We therefore recommend that you have a look at the source code of the accompanying example program. Usually the quickest way of learning about programming is to examine and adapt a ready-made program. We've used Delphi for the example program because this language is very easy to follow. The source code could easily be converted into C++, for example. Elsewhere in this issue you'll find a simple COM Port Tester, which can be used to check the operation of your own programming efforts.

(020388-1)

Automatic Fridge Switch for Caravans

024007-1,
December 2002, p. 88

Contrary to what is indicated by the circuit diagram, R4 should be 330 Ω, while R6 is better changed to 1 kΩ. Regarding the adjustment, turn P1 until the relay pulls in at 13.5 V. Next, P2 is adjusted until the relay is de-energised at 11.5 V (i.e., do not use P1 for this adjustment).

Direct Current Dimmer

024041-1,
July/August 2002, p. 100

The text erroneously states that the 555 oscillates at 65 Hz. This should be corrected to read "65 ms (15 Hz)". To enable the 555 to oscillate at 65 Hz, the value of capacitor C2 should be lowered to about 220 nF.

Digital RF Wattmeter with LC Display

020026-1, October 2002, p. 16

Capacitor C2 is missing from the circuit diagram. It should be connected between junction C1/R6/R7 and pin 8 (INP) of IC1. The capacitor at pin 1 of IC1 should be labelled C3, not C2.

IR Receiver for the I²C Bus

024033-1,
December 2002, p. 85

In the upper right hand part of the circuit diagram, the names of the I²C bus lines have been transposed and should be corrected as follows:

SDA on pin 15 of IC1
SCL on pin 14 of IC1
$\overline{\text{INT}}$ on pin 19 of IC2

It should be noted that the SAA3049 is no longer produced by Philips Components.

USB Driver Programming

020109-1/2,
October and November 2002

In some cases, the chip will not respond and all ports will remain low. Although Windows has detected that a new USB device has been connected, this will be not recognized. The problem is solved by fitting pull-up resistors R3 and R2 as indicated in Figure 1. These resistors should not be omitted.

Telephone Baby Monitor

012016-1,
November 2002, p. 46

In the circuit diagram, the value of C8 should be corrected to read 470 pF.

VCP2002 Video Copy Processor

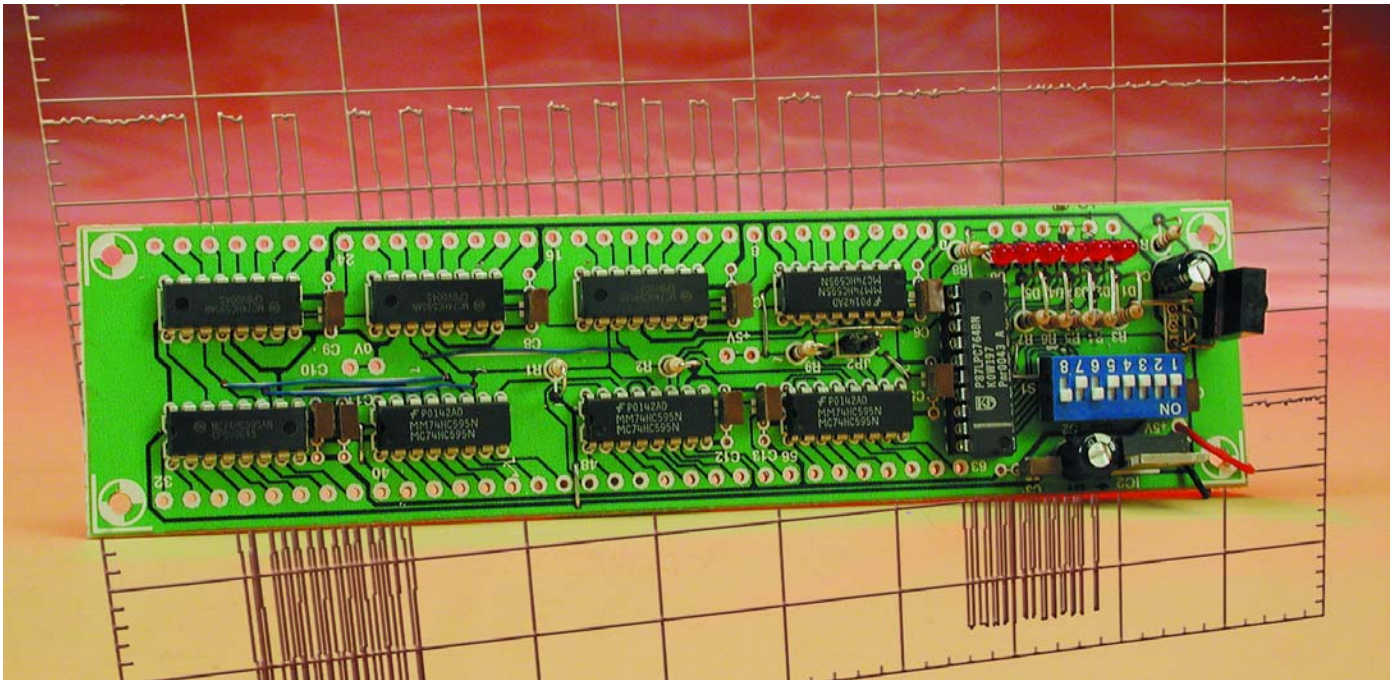
010121-1,
July/August 2002, p. 12

On some printed circuit boards the silk screen printing for the six wire links is missing. These links are, however, indicated in the PCB component overlay printed in the magazine.

SAA3049 Replacement

a microcontroller-based RC5 decoder

Sadly enough, the popular Philips SAA3049 remote control decoder IC is no longer available. No semiconductor foundry offers a replacement type, probably because nowadays remote-control decoding in consumer electronics is performed by customer-specific microcontrollers. We can do the same ourselves!



The number of ICs that have been specifically developed for decoding infrared messages is significantly less that the number of available transmitter ICs. This is because there is often already a microprocessor in the receiving device, and it can handle the decoding 'on the side'. Depending on the application, a certain amount of supplementary logic is always necessary to integrate a decoder IC into a circuit, and this contributes to cost and space problems. Modern microcontrollers can handle this task just as well or better, and they only cost around 50 pence each in large quantities.

The SAA3049, which has long been a popular IC for decoding RC5 and RECS80 remote control signals, is also based on a microcontroller, and it needs only a 4-MHz crystal and an infrared receiver as external circuit elements. However, some additional components, such as flip-flops, are often needed to handle the switching tasks necessary to enable the remotely controlled device to process the decoded data. Besides this, the SAA3049 is not exactly blessed with an abundance of outputs.

Philips have now deleted the SAA3049 from their product line.

Features

– Decodes RC5 and RECS80 signals
– Configurable receiver address
– Addressing options: the address function can be disabled if desired, allowing the circuit to respond to all addresses
– Commands can be output in binary or decimal (reduced) format
– Analysis function for transmission protocol
– Six direct, switched outputs available
– Switched outputs can operate in latched or momentary mode
– Serial output for connecting up to 64 outputs
– Command Enable signal available
– No oscillator circuitry necessary
– No reset circuitry necessary

The receiver / decoder circuit presented here works with the following transmitter ICs:

RC5 code: SAA3006, SAA3010 (Philips)
HT6230 (Holtek)
PT2211 (Princeton)

RECS80 code: SAA3004, SAA3007, SAA3008 (Philips)
M3004, M3005, M3006 (ST Microelectronics)

This makes it very difficult for non-industrial users who do not have any programming experience to put together remote-control systems. In the process of developing an alternative to the SAA3049, we can also take the opportunity to make a few 'incidental' improvements that eliminate some of its drawbacks:
– no need for a 4-MHz crystal
– no need for extra circuitry to handle switching tasks
– no need for external reset circuitry
– more outputs

An existing remote control unit for a television set, video recorder or stereo system can serve as the transmitter, or a 'bargain bin' unit can be used. If you don't know for sure which format your transmitter sends, we recommend building the remote control analyser described in the October 2001 issue of *Elektor Electronics*.

The microcontroller

The SAA3049 replacement is based on an 87LPC764 microprocessor. This IC is a member of the 8051 family with 2 to 4 kB of OTP ROM program memory. The designation 'LPC' in the type number stands for 'low price, low power and low pin count'. Besides rapid instruction processing, it features an on-board RC oscillator and reset logic. This saves pins and gives the user access to up to 18 I/O lines.

The internal RC oscillator clocks the CPU at approximately 6 MHz, so one cycle lasts 1 µs. This is fast enough to reliably read RC5 codes. The tolerance range for the oscillator frequency is the reason why all of the times specified for this circuit can vary by as much as ±20%.

A summary of the major features of the microcontroller is provided in the '87LPC764 Short-Form Specifications' box. Additional specifications and the complete data sheet can be downloaded via the Internet from the Philips website at www.semiconductors.com/pip/P87LPC764.html. The complete data sheet can be found at www.semiconductors.com/acrobat/datasheets/87LPC764_10.pdf.

The software

In developing the software, special attention was given to dependable decoding. In accordance with the motto 'everything is possible with software', suitable programming routines are used to ensure that commands are recognised and extraneous protocols and interference are filtered out. For instance, the following checks are performed for the RC5 code:
– pulse length verification
– multiple signal sampling
– the two half-bits are different
– command value less than '64'
– matching of message address and configured address

It is normally not particularly difficult to read a remote-control message. However, if the microcontroller is clocked by an RC oscillator, the frequency of the oscillator may vary over a range of ±20%. An additional source of timing errors is pulse length corruption from the photoreceiver module, as can be seen using the Vishay TSOP12xx module as an example. The chart in **Figure 1** shows the possible range of variation of the output pulse length as a function of radiated signal strength, for a test signal that is switched on and off for exactly 600 µs. The timing corruption can be as large as 100 µs, which is quite significant and must be taken into account by the software.

It is thus not sufficient to simply synchronise to the signal at the start of the message and then sample the signal level at fixed intervals for the rest of the message. With such a

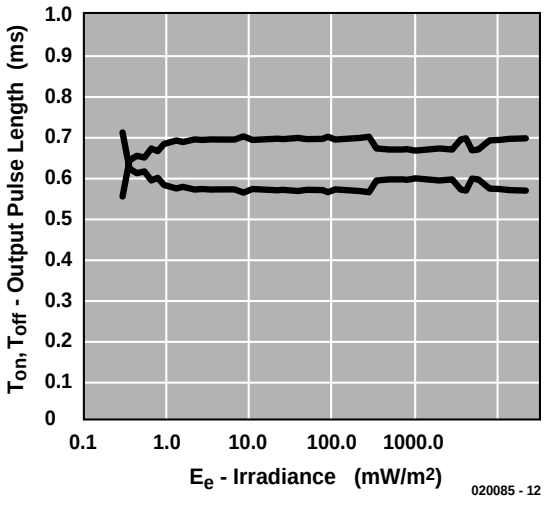


Figure 1. Corruption of the pulse length of a test signal, as measured at the output of a photodetector module.

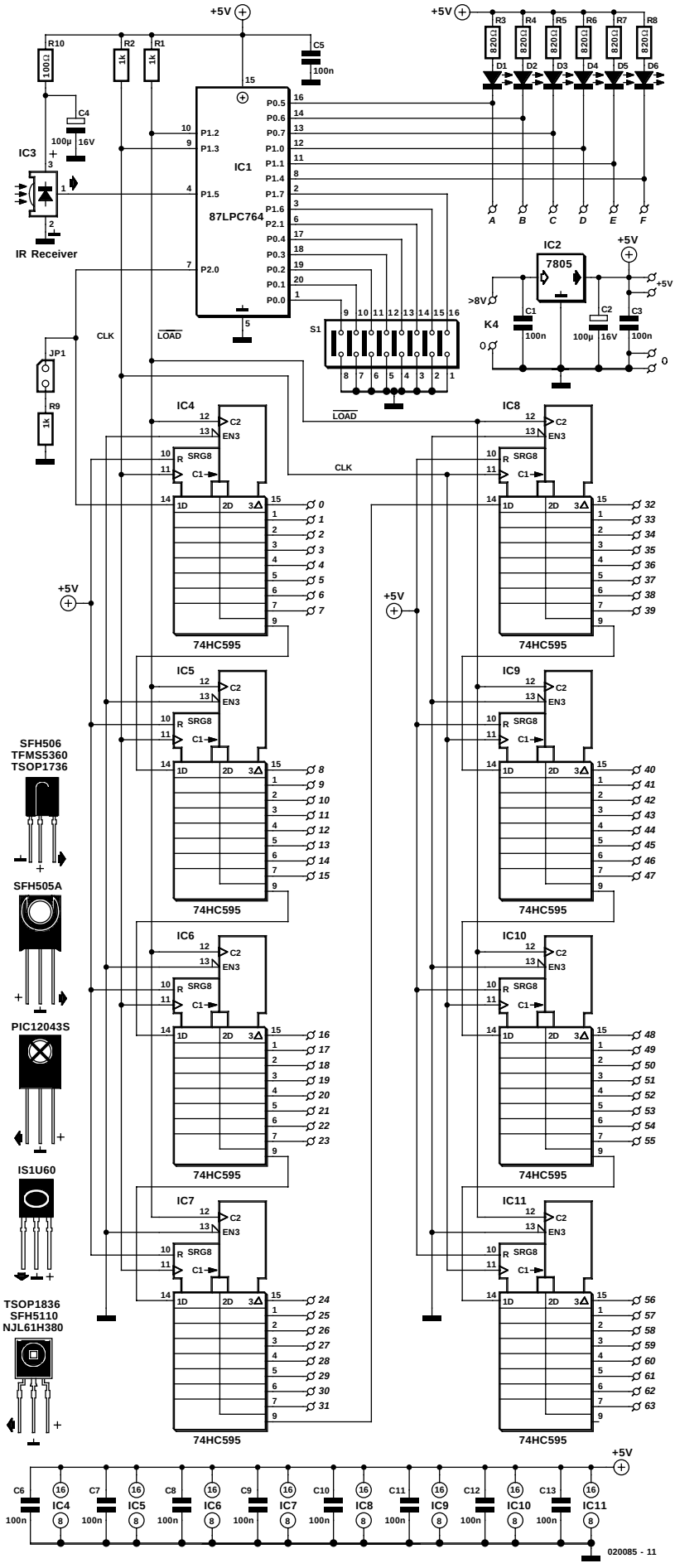


Figure 2. The actual RC5 receiver consists of the IR module, the microcontroller and the DIP switches. The shift registers are optional.

scheme, the sampling times would be totally false by the time the middle of the message was reached, if not sooner. For this reason, the software resynchronises to individual bits several times during the message, in order to circumvent problems arising from timing variations.

The circuit

At first glance, the circuit of the SAA3049 replacement shown in **Figure 2** appears to be rather extensive. However, on closer examination it proves to consist of little more than an IR receiver, a microcontroller and a set of serial-to-parallel converters providing up to 64 switched or pulse outputs.

Infrared reception

The infrared radiation, which is modulated using a 36 or 38 kHz carrier frequency, is received by the photodetector module IC3. Several different types of IC can be used here, but the pinout of the type used must be carefully observed when fitting it to the board.

IC3 is a miniaturised receiver that can be used somewhere in the frequency range of 30–50 kHz, depending on the particular type employed. The receiver diode and amplifier stages are integrated into a single IC, so no external circuitry is necessary. The demodulated signal can be directly processed by a microcontroller. The special features of this IC are reliable operation, even in a noisy environment, and suppression of undesired output pulses.

The coded signal is fed directly (in inverted form) to the microcontroller (IC1) for decoding. R10 and C4 form a low-pass network that isolates the receiver IC from noise on the supply line. These components are essential, since the IC responds to noise spikes with a drastic reduction in sensitivity.

Setting the address

The address to which the chip is

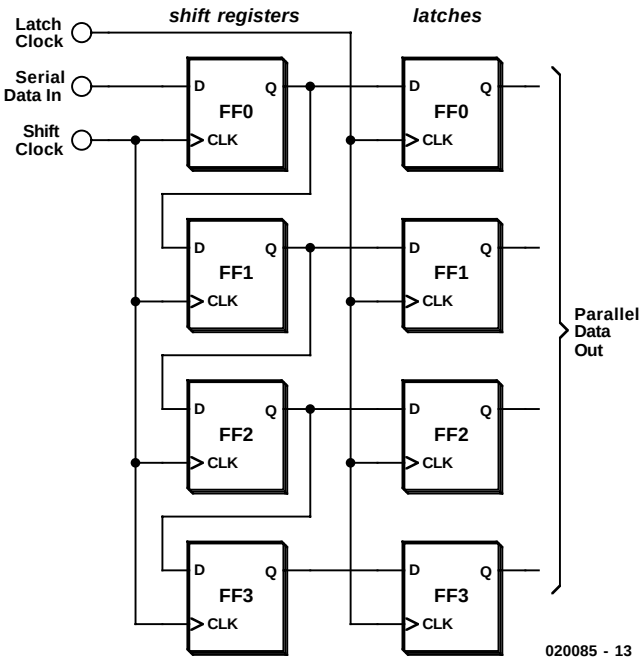


Figure 3. The 74HC595 consists of two banks of eight cascaded flop-flops.

87LPC764 Short-Form Specifications

– Supply voltage: 2.7–6.0 V
– Instruction cycle: 300–600 ns / 20 MHz
– RAM: 128 bytes
– ROM: 2 or 4 kBytes (87LPC764)
– 2 timers
– 2 analogue comparators with integrated 1.23-V reference
– Full-duplex UART
– I²C communication port
– 2 external interrupts and 8 keypad interrupts
– Watchdog with separate oscillator
– On-chip power-on reset
– Undervoltage reset
– Internal RC oscillator with no external circuitry, or crystal oscillator with capacitors
– Programmable port configuration, quasi-bidirectional: – open drain, – push-pull, – input only
– Serially programmable
– Idle and power-down modes

supposed to respond, which is contained in the message, is set using pins P0.0 through P0.4. The five DIP switches can select up to 32 different addresses. An open switch corresponds to a logic '1', since the microcontroller has internal pull-up resistors. If all of the switches are closed, all of the pins are tied to ground (address 00000). Since the RECS80 code defines only seven addresses, only the levels on pins P0.0 through P0.2 are evaluated for this code.

Command enable
Port pin P1.2 will be active (Low) for approximately 65 ms when a valid message has been read. This signal can trigger subsequent peripheral circuitry, for example if the binary command data are evaluated by subsequent logic. However, this function can also be utilised to visibly indicate receipt of a message, by means of an LED. This pin is configured as an open-drain output, so it may be necessary to connect a pull-up resistor, for instance if the signal is to be displayed on an oscilloscope.

Settings

The operating mode can be programmed via one jumper and two DIP switches connected to port pins.

P2.0: Code select
In the case of the SAA3049, the circuitry precisely controls how the IC responds to the two codes (RC5 and RECS80). With the original decoder IC, this is communicated using a dedicated code-select pin, but the replacement design must resort to software tricks, since it does not have any spare pins. Here port P2.0, whose normal function is to provide serial output data to the subsequent hardware, must perform a dual role.

When the program is started, this pin is switched to the input configuration with an internal pull-up resistor. The software then polls the level on the pin. If the jumper is fitted, the level will be Low, selecting the RECS80 mode. If the jumper is not present, the level will be High, selecting the RC5 mode. After the level has been read, the pin is switched to the output configuration, using the push-pull option to prevent R9 from affecting the logical data levels.

P1.7: Switched / binary output
This pin also has an internal pull-up resistor. With a High level, the switched output mode is enabled, causing the output pins to respond directly to commands 0 through 5 of the infrared remote control unit. This gives the following relationship:

Command 0: Pin P0.5 (A)
Command 1: Pin P0.6 (B)
Command 2: Pin P0.7 (C)
Command 3: Pin P1.0 (D)
Command 4: Pin P1.1 (E)
Command 5: Pin P1.4 (F)

Pressing a button thus changes the logical state of the associated pin.

If a logical Low level is present on P1.7, the binary value of the command is made available on outputs A–F, similar to the outputs of the SAA3049.

P2.1: Latched / momentary output
This pin also has an internal pull-up resistor. When the applied level is High (P2.1 open), the outputs operate in latched mode. Otherwise they change states for only approxi-

mately one second after a button press and then return to their original states (High by default). This corresponds to monostable operation. If a button is held depressed, the associated output stays Low as long as the button remains pressed.

COMPONENTS LIST

Resistors:
R1,R2,R9 = 1kΩ
R3-R8 = 820Ω
R10 = 100Ω

Capacitors:
C1,C3,C5 = 100nF
C2,C4 = 100µF 16V radial

Semiconductors:
D1-D6 = LED, red, low current
IC1 = 87LPC764BN, programmed, order code **020085-41**
IC2 = 7805
IC3 = IR receiver module SFH5510, PIC2604SM, TFM5360 or similar
IC4-IC11 = 74HC595

Miscellaneous:
JP1 = jumper
S1 = 8-way DIP switch
PCB, order code **020085-1**
Disk, source and HEX code, order code **020085-11** or Free Download

Applications

Among the multitude of potential applications, we can choose four examples for examination.

A simple remote control system
Here address pins P0.0–P0.4 are left open, so the address of the remote control unit does not matter. Each time one of the buttons 1 through 6 is pressed, the corresponding output changes state. The CA signal on pin P1.2 is activated each time a command is received. The outputs cannot source more than 1 mA, but they can sink up to 20 mA in the Low state.

Address / command analyser
To determine which RC5 address a remote control unit transmits and the relationship between the commands and the buttons, first connect the coding input (P1.6) to ground. Leave all of the address inputs open. The address and command data will be output in binary form. When the program is started, a check will be made to see whether the address inputs are actually open. They cannot be checked any more after this, since it will only be possible to read the states of the output latches, and they will be set to the value of the output address.

The output levels can be checked using an oscilloscope, voltage tester or logic tester. In the analysis mode, all of the outputs operate in latching mode, regardless of the settings of the other configuration inputs. In no case may any of address inputs P0.0–P0.4 be connected to ground in this operating mode, since this could produce a short circuit. All relevant pins are active Low. The combination of address 4 (binary 00100) and command 7 (binary 000111) thus yields the following levels:

P0.0 = 1	address
P0.1 = 1	
P0.2 = 0	
P0.3 = 1	
P0.4 = 1	
P0.5 = 0	command
P0.6 = 0	
P0.7 = 0	
P1.0 = 1	
P1.1 = 1	
P1.4 = 1	

Door opener
In this application, a switched output is to be momentarily activated when a button is pressed. This requires the outputs to be operated in the retriggerable monostable mode.

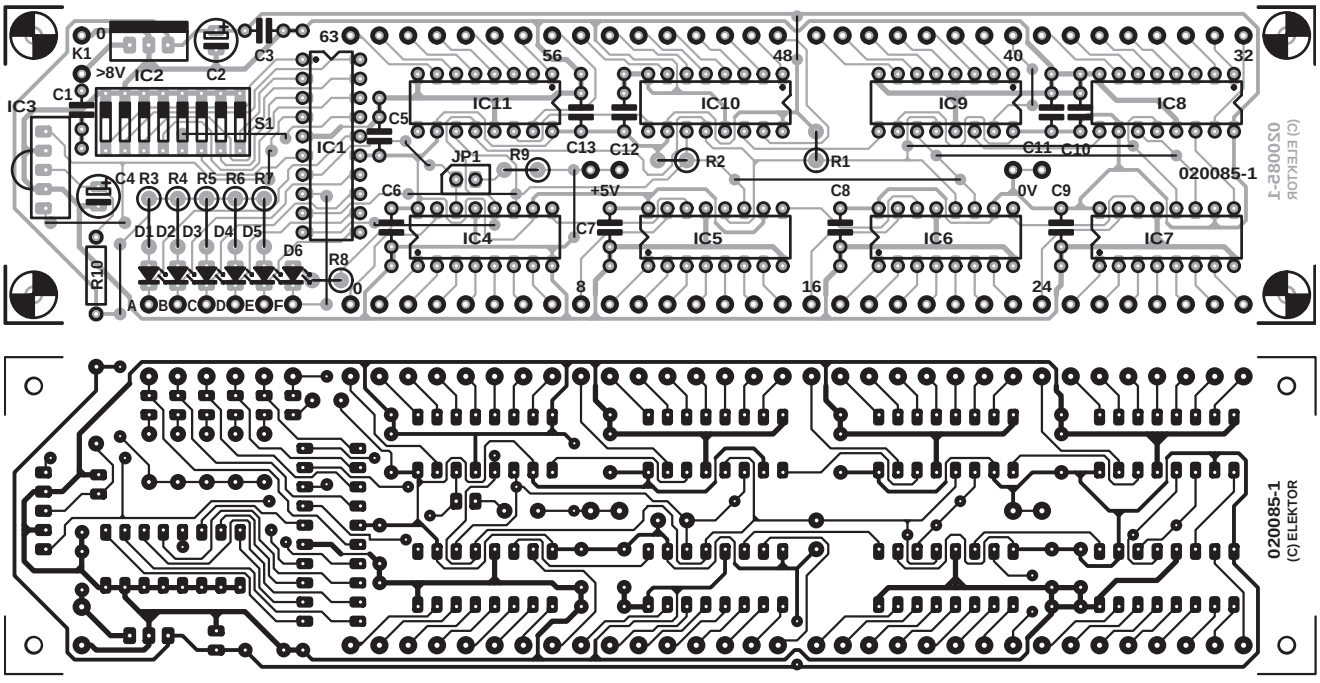


Figure 4. The circuit board layout is uncomplicated.

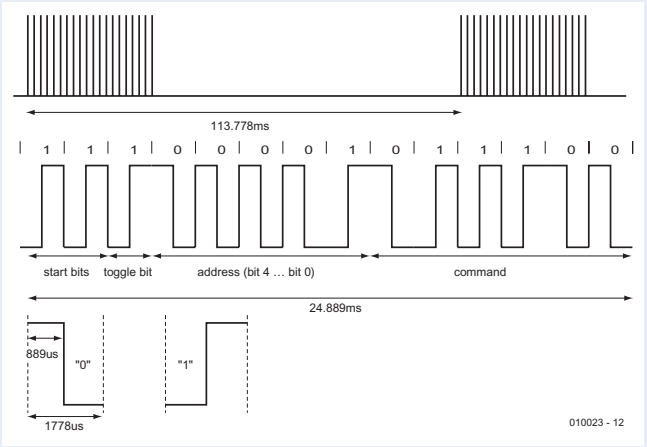
Remote Control Code Specifications

The RC5 code

The RC5 standard, which is widely used in Europe for infrared data transmission, was developed by Philips. This code can transfer 2048 different commands, divided into 32 addressable groups of 64 commands each. In this system, each device is assigned its own address, so only the desired device responds to a particular command (for instance, the loudness of your stereo system will not be affected when you adjust the loudness of your television set). The transmitted code consists of a 14-bit data word organised as follows:

– 2 start bits for setting the AGC (automatic gain control) level of the receiver IC
– 1 toggle bit to mark the start of a new command
– 5 system address bits
– 6 command bits

The value of the toggle bit changes each time a new button is pressed, to make it possible to distinguish between pressing the same button again and holding a button pressed. The toggle bit is followed by five address bits, which indicate which device should respond to the command. Finally, the actual command is transmitted. With the RC5 code, the commands are bi-phase coded, with each bit consisting of two half bits having opposite values. The combination Low/High marks a set bit ('1'), while the combination High/Low marks a reset bit ('0'). The period of each bit is 1.776 ms, which yields a total period for one message of 24.889 ms. The RC5 code is used by many manufacturers of entertainment electronics equipment, including Loewe, Philips, Grundig and Marantz.



RC5 code specifications

RC5 device codes

System address	Binary	Device
0	00000	TV1
1	00001	TV2
2	00010	Videotext
3	00011	Extension for TV1 and TV2
4	00100	Laser Vision player
5	00101	Video recorder 1 (VCR1)
6	00110	Video recorder 2 (VCR2)
7	00111	Reserved
8	01000	SAT1
9	01001	Extension for VCR1 and VCR2
10	01010	SAT2
11	01011	Reserved
12	01100	CD-Video
13	01101	Reserved
14	01110	CD-Photo
15	01111	Reserved
16	10000	Audio preamplifier 1
17	10001	Tuner
18	10010	Analogue cassette recorder
19	10011	Audio preamplifier 2
20	10100	CD
21	10101	Audio rack or recorder
22	10110	Audio satellite receiver
23	10111	DCC recorder
24	11000	Reserved
25	11001	Reserved
26	11010	Writeable CD
27–31	.....	Reserved

Selected commands

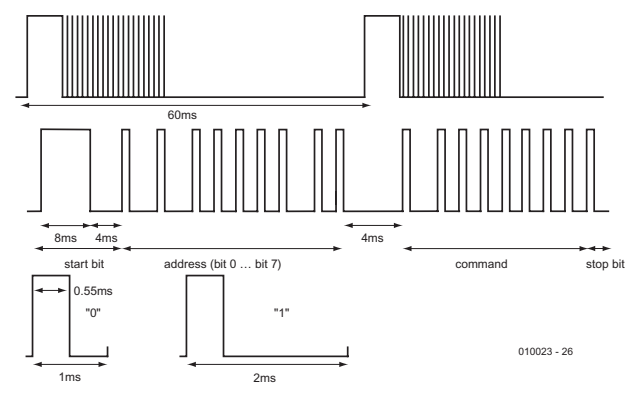
Command	Meaning
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
16	Volume +
17	Volume –
18	Brightness +
19	Brightness –
20	Colour saturation +
21	Colour saturation –

The RECS80 code

The Philips RECS80 code employs pulse-distance modulation for data transmission, which is characterised by the fact that a High pulse of a defined length is followed by a pause having a variable length, with the duration of the pause denoting the logical state of the transmission bit. There are 1280 defined code sequences, divided into 64 commands for each of 20 subsystems. Subsystems are defined to be various types of command receivers, such as television sets, video recorders and so on. In combination with one toggle bit and one start bit, this yields a maximum word length of 12 bits.

For historical reasons, a data word of only 11 bits (containing three subaddress bits) is transmitted for subaddresses 1 through 7, which complicates the decoding software. The toggle bit is generated by a counter in the transmitter IC that is incremented each time a button is pressed, but not if a button is held pressed. Consideration must be given to the fact that two toggle bits are only used for data transmission in the Flash mode For ICs that transmit

modulated infrared light, the first toggle bit takes the form of a start bit with a fixed length.
The RECS80 protocol codes individual bits using pauses of different durations following light pulses having a fixed duration (140.8 µs). The time between successive light pulses, which determines the bit value, is 5.06 ms for a '0' and 7.60 ms for a '1' (assuming that a 455-kHz resonator is used in the transmitter). Although the length of time required to transmit a data packet varies depending on the transmitted command, the total interval until the transmission of the next signal (including a pause) is fixed at 121 ms. The modulation frequency is 38 kHz.



RECS80 code specifications

P1.7 remains open, so the switched mode is enabled. Pin P2.1 is tied to ground to select the momentary output mode. This causes the selected output to be active for approximately one second. If the button on the remote control unit is held pressed, the output level remains active for a correspondingly longer time. The output is Low when active, returning High after the timeout expires.

64-channel remote control
In order to realise the greatest possible number of switched outputs with a minimum amount of wiring, the microprocessor must be fitted with suitable external hardware and matching software. Here the choice fell on the inexpensive and readily available 74HC595 serial-to-parallel converter.

This IC, whose operating principle is shown in **Figure 3**, consists of 16 cascaded flip-flops arranged in two banks. The data are fed in serially to FF0 of the first bank. The shift clock signal CLK transfers the data to the outputs of each of the flip-flops and thus to the inputs of the following flip-flops in the chain. After a certain number of SCL pulses, which is determined by the user, the Latch Clock pin (LCL) is clocked once to make the originally serial data available in parallel form on the outputs of the second bank of flip-flops.

The 74HC595 also has several other useful features, as described below.

Reset
A Low level on the RESET input resets the internal logic (outputs disabled, all FFs set to zero). This feature is not used here, since the outputs can be disabled via the OUPUT ENABLE pin. After a microcontroller reset, all of the registers are anyhow filled with zeroes.

Latch
Each output can source and sink up to 35 mA. This is easily enough to switch connected loads such as LEDs or transistors, which in turn can drive higher-power relays. The latch is enabled by a Low level on the OUPUT ENABLE pin, and its outputs retain their levels indefinitely.

Latch Clock
A rising edge on this pin transfers the data from the shift register to the latch. This may only be initiated after 64 clock pulses have been applied to clock in the data.

Only as many 74HC595 ICs as necessary need be fitted to the circuit board. For instance, if you do not need more than 24 outputs, you only have to fit three ICs. The software will issue 64 clock pulses for each new command, so the data will automatically be properly positioned in the appropriate ICs. All that matters is that the ICs are wired in series with gaps in between.

The clock output of the microcontroller (P1.3) is configured as an open-drain output, which means that it needs a pull-up resistor. The 64 outputs are available with either latching or momentary operation, depending on the selected mode.

PCB construction

The layout of the printed circuit board is shown in **Figure 4**. The receiver and decoder are located at the far left. If all you want to build is a simple remote control system, as previously described, you can saw off the portion of the circuit board to the right of the microcontroller IC. However, in this case JP1 cannot be fitted, so the circuit will only work with RC5 signals. Microcontroller outputs P0.5–P1.4 are routed to the solder pins located directly below the row of LEDs.

The optional shift register ICs are arranged in a loop running from left to right and then back, with outputs 0–31 at the bottom edge of the board and outputs 32–63 at the top edge. Since the printed circuit board is single-sided, several wire bridges are necessary. Some of them run underneath the ICs, which is another good reason for using IC sockets.

In the schematic diagram, you can see that it is possible to use several different types of IR receiver, not all of which are pin compatible. The circuit board layout has been designed such that all of these types can be fitted, but you must be careful to fit the leads correctly.

(020085-1)

References

IR Remote Control Codes, Elektor Electronics, March–April 2001

IR Code Analyser, Elektor Electronics, October 2001

Model Remote Control, Elektor Electronics, May 2001

Learning RC5 Control Decoder, Elektor Electronics, January 2001

Multi-standard Infrared Receiver, Elektor Electronics, April 2002

Infrared Transceiver for the PC, Elektor Electronics, January–February 2002

From PC Speakers to Intercom

By A. C. J. Bauer

Most loudspeaker systems supplied with inexpensive multimedia PCs are of a doubtful quality and quickly exchanged for 'something better'. When the new kit is installed the low-fi stuff is banned to junior's bedroom, the loft or some other storage space.

This article shows how to turn such cheap loudspeaker systems into a simple but effective and above all cheap intercom system.

In nearly all cases, one of the loudspeaker boxes contains a power supply, a stereo amplifier and a loudspeaker. The other box is a 'slave' and contains nothing but a loudspeaker.

Intercom system

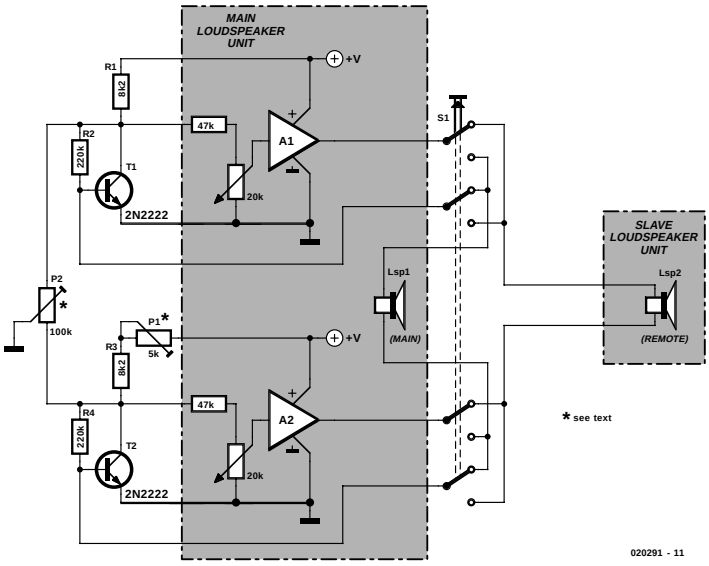
A recipe for turning a set of cheap PC loudspeakers into an intercom is shown in **Figure 1**. Only the parts outside the shaded box need to be added. The slave loudspeaker acts as a microphone when the multi-pole change-over switch on or near the main unit is pressed. Some amplification is then required, hence the addition of transistors T1 and T2. Connecting the two amplifiers in the main unit in a balanced configuration effectively reduces the noise in the (long) cable between the intercom stations. Trimpot P1 is used to set the exact balance between the amplifiers when a relatively long line is used. P2 is optional and may only be needed in cases where extremely high noise levels are present.

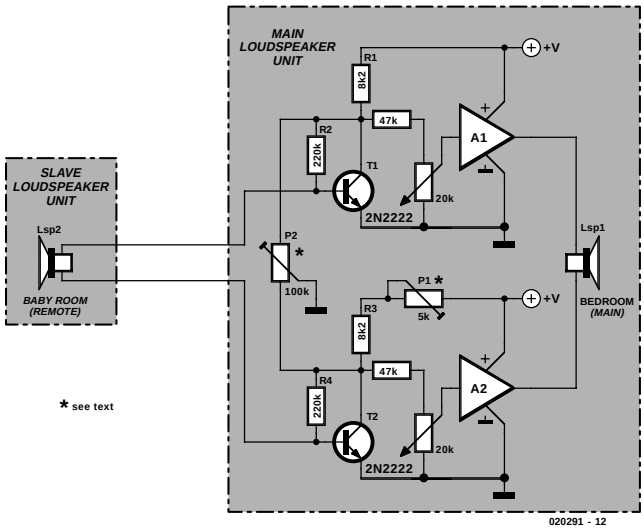
In most, but not all, PC loudspeaker units, the on/off switch switches the positive supply voltage rather than the mains voltage. That allows the +V rail for the transistor stages to be taken from a terminal on the on/off switch.

Babyphone

For a babyphone it is not normally required to 'talk' to the remote station and the circuit may be modified to the one shown in **Figure 2**. If more sensitivity is required, the 47 kΩ resistors may be shorted out.

(020291-1)





Cooling Fans

a backgrounder on fan selection

Source: Sunon, Taiwan

The total cooling requirements are critical to operate any heat-wasting system efficiently. A good way to prevent huge repair bills is to provide operating conditions that maximize the performance and life of all components in the system.

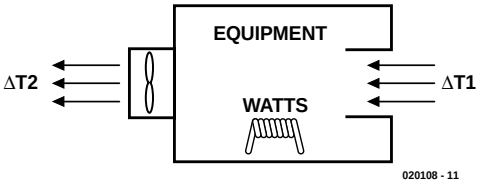


Figure 1. Heat generated in electrical equipment may be reduced by a sufficient amount of internal airflow.

When making the selection of the fan motor for ordinary use, the following methods are used.

1. Determine the amount of heat generated inside the equipment.
2. Decide the permissible temperature rise inside the equipment.
3. Calculate the air volume necessary from equation.
4. Estimate the system impedance in the unit.
5. Select the fan by performance curve shown in the manufacturer's catalogue or data sheet.

The volume of air flow required to cool an equipment can be determined if the internal heat dissipation and the allowable total rise in temperature are known, see **Figure 1**.

The Basic Heat Transfer equation states:

$$H = C_p \times W \times \Delta T$$

where

H = Amount of heat transferred;
C_p = Specific heat capacity of air;
ΔT = Temperature rise within the cabinet;
W = Mass flow.

Obviously we have W = CFM x D, where D = Density, CFM = cubic feet/minute.

By substitution, we obtain

$$Q(CFM) = \frac{Q}{C_p \times D \times \Delta T}$$

By incorporating conversion factors and specific heat capacity and density for air at sea level, we arrive at the heat dissipation equation:

CFM = 3160 x kilowatts / Δ°F

Then, we obtain the following equations:

$$Q(CFM) = \frac{3.16 \times P}{\Delta Tf} = \frac{1.76 \times P}{\Delta Tc}$$

where
Q = Required air flow;
P = Internal heat dissipation;
T_f = Allowable temperature rise in °F;
T_c = Allowable temperature rise in °C;
$\Delta T = \Delta T_1 - \Delta T_2$.

The conversion of Temperature vs. Air Flow is summarized in **Table 1**.

Examples, please!

Example 1:
If the internal heat dissipation is 500 watts and ΔT is 20°F, we get the following result:

$$Q(M^3 / Min.) = \frac{0.09 \times P}{\Delta Tf} = \frac{0.05 \times P}{\Delta Tc}$$

or

$$Q = \frac{3.16 \times 500\,(watts)}{20} = 79\,CFM$$

Example 2:
If the internal heat dissipation is 500 watts and ΔT is 10 °C:

$$Q = \frac{0.09 \times 500\,(watts)}{20} = 2.25\,M^3/Min$$

or

$$Q = \frac{1.76 \times 500\,(watts)}{10} = 88\,CFM$$

The Total System Resistance / System Characteristic Curve

While air is moving, its flow will encounter resistance of other components in the system along the path of the air stream. This imped-

ance restricts free air and passage of air. The resulting change in pressure (ΔP) is the static pressure measured in inches of water (Inch-H_2O).

In order to specify the cooling per slot in watts, the system designer/manufacturer should not only have a valid air flow curve to determine the maximum air flow, but he/she should also know the system air resistance curve. There is a loss of air pressure due to resistance of components inside the enclosure. This loss varies with air flow and is known as **system resistance**.

The System Characteristic Curve formula is:

$$\Delta P = K\Omega^n$$

Where
K = system characteristic constant;
Q = air flow, CFM;
N = turbulence factor, $1 \leq n \leq 2$;
(laminar flow, n = 1;
turbulent flow, n = 2).

The intersection point of system characteristics curve and air performance curve of an air moving device like a fan is named **System Operating Point**. As shown in **Figure 2**, this point describes the best air moving device for your application. At this point, the change in slope of the air performance curve is minimized while the change in slope of the system characteristics curve is at its lowest. Note that the static efficiency (air flow times static pressure divided by power) is also optimised.

Design Considerations

1. Keep the air flow path as unobstructed as possible. The air intake and outlet should be kept free for all air flow.
2. Guide vertical air flow through your system, it will assure the flow moves more smoothly and increase cooling efficiency.
3. If a filter is required, you should consider the additional resistance to air flow.

Next, we'll discuss some examples of selecting "the" best fan for your application.
Example 1. **Figure 3** is an air performance curve of a typical Sunon DC Cooling Fan with size 60 × 60 × 25 mm. The fan might be, for example, applied at Point A or Point C, delivering 6 CFM or 20 CFM respectively, if the system resistance were imposed a pressure drop of 0.16 (Point A) or 0.04 (Point C) inch-H_2O on the air stream. If the system can be modified to apply at Point B, the fan might be delivering more than 12 CFM at a pressure of only 0.09 Inch-H_2O.

Example 2. As shown in **Figure 4**, Curve 2 describes a fan of the same size and configuration but lower speed than Curve 1. If the system requires only 15 CFM at 0.05 inch-H_2O, the pressure drop/flow rate parabola is through Point B. Therefore, a fan that provides an air flow of 18 CFM at zero static pressure is adequate for cooling. Thus, the final arrangement is to use a fan of lower speed. Figure 4 in one graph summarizes the change from one fan to the other. In some cases, of course, it may even be possible to move to a physically smaller fan with the same air flow, if system resistance is sufficiently reduced.

Example 3. **Figure 5** shows the air performance curves of 40 × 40 × 6 mm (Curve 3), 30 × 30 × 6 mm (Curve 2) and 25 × 25 × 6 mm (Curve 1) DC fans running at middle speed.

Case 1: If the system acquires a system resistance of 0.025 inch-H_2O and requires an air flow of 2 CFM to cool off, a 40 × 6 mm DC fan is rec-

Table 1. Conversion of Temperature vs. Air Flow

KWh		0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
ΔT °C	ΔT °F										
50	90	18	35	53	70	88	105	123	141	158	176
45	81	20	39	59	78	98	117	137	156	176	195
40	72	22	44	66	88	110	132	154	176	195	220
35	63	25	50	75	100	125	151	176	201	226	251
30	54	29	59	88	117	146	176	205	234	264	293
25	45	35	75	105	141	176	211	246	281	316	351
20	36	44	88	132	176	220	264	308	351	396	439
15	27	59	117	176	234	293	351	410	469	527	586
10	18	88	176	264	351	439	527	615	704	791	879
5	9	176	351	527	704	879	1055	1230	1406	1582	1758

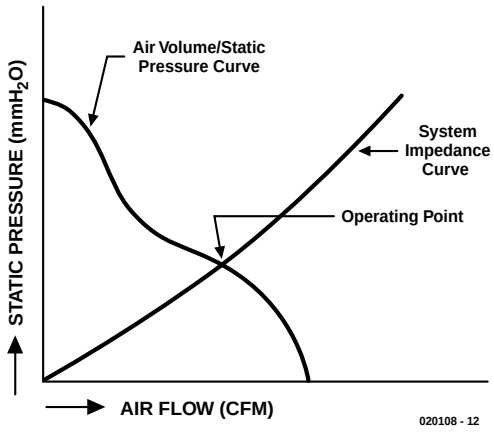


Figure 2. The intersection of the air volume/static pressure curve with the system impedance curve is called the Operating Point.

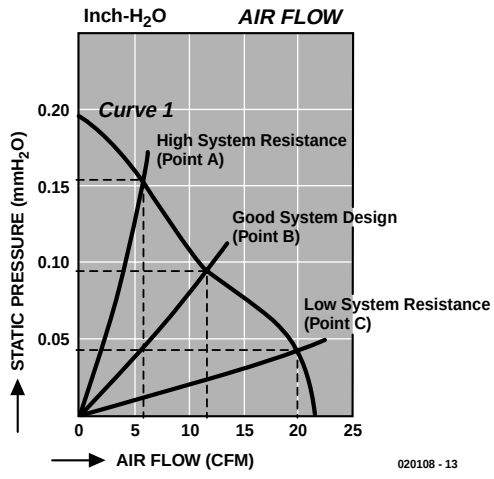


Figure 3. Air performance curve of 60 x 60 x 25 mm fan at middle speed.

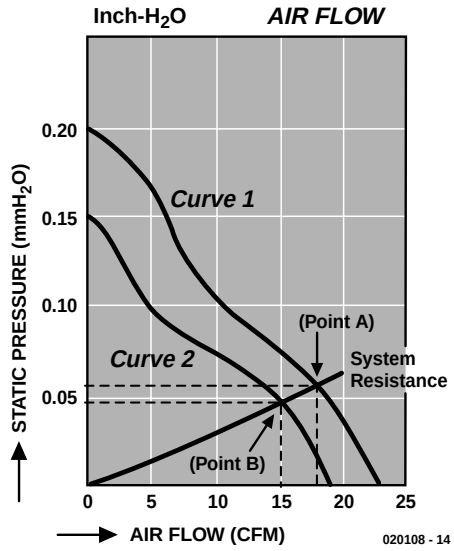


Figure 4. Air performance curve of 60 x 60 x 15 mm fan at low and middle speed.

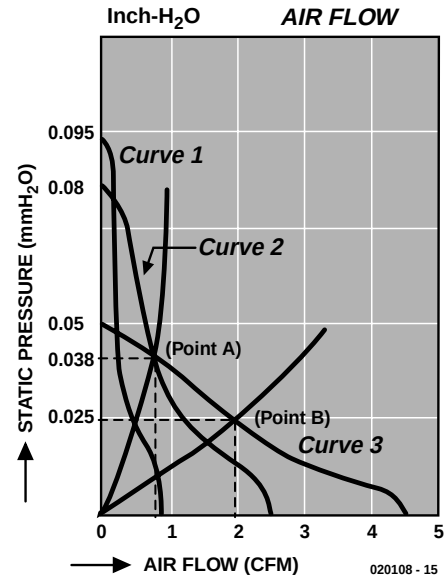


Figure 5. Air performance curve of 40 x 40 x 6 mm, 30 x 30 x 6 mm and 25 x 25 x 6 mm fans.

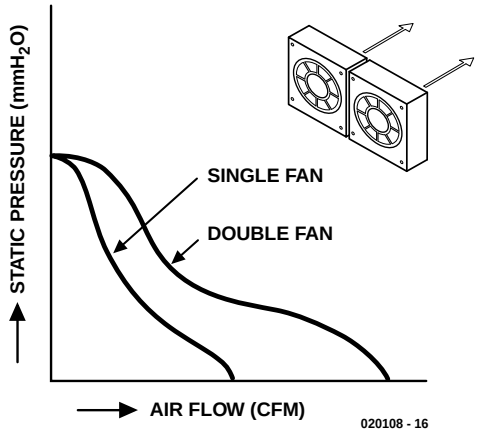


Figure 6. Performance of parallel-mounted fans vs. single fan.

ommended (please refer to the Operation Point B).

Case 2: If there are more components added to the system and/or there is a more compact physical reconfiguration, the result will be a higher system resistance. Now, assuming that the system resistance is increased to 0.038 inch-H_2O and requires 0.85 CFM to cool off the system, then there are two fans, 40 × 6 mm and 30 × 6 mm, available for selection (please refer to Operating Point A). Another excellent option for cooling a system with a high system resistance is the Sunon Micro DC Blower.

Parallel and Series Operation

Parallel operation is defined as using two or more fans side by side — see **Figure 6**.

The air flow volume produced by two fans in parallel will be double in the (hypothetical) free-air condition only. In practice, a smaller increase in flow results is achieved as a result of the high system resistance of the enclosure. Thus, this type of application is only recommended for low system resistance situations — i.e., when the fans can operate at near-free air delivery.

Series operation is defined as using two or more fans in series, see **Figure 7**. The static pressure capacity of two fans in series can be doubled at zero air flow condition, but does not increase the airflow in the free-air situation. An additional fan in series increases the volume flow in a higher static pressure enclosure. Consequently, series operation of fans yields best results in systems with relatively high air flow resistance.

Acoustic noise level

Acoustic sound measurement of Sunon fans is made in an anechoic room with background noise less than 15 dBA. The measured fan is running in free air with a microphone at a distance of one meter from the fan intake.

Sound Pressure Level (SPL), which is environmentally dependent, and Sound Power Level (PWL) are defined as following :

$$SPL = 20 \log_{10} P/P_{ref}$$

and

$$PWL = 10 \log_{10} W/W_{ref}$$

where

P = Pressure
P_{ref} = A reference pressure
W = Acoustic power of the source
W_{ref} = An acoustic reference power

Fan noise data is usually plotted as SPL against the octave frequency bands. The following provides an indication of the effect of dBA changes:

3 dBA = barely noticeable
5 dBA = noticeable
10 dBA = twice as loud

Noise levels (A-weighted human perception):

0 to 20 dBA	very faint
20 to 40 dBA	faint
40 to 60 dBA	moderate
60 to 80 dBA	loud
80 to 100 dBA	very loud
100 to 140 dBA	deafening

The following five guidelines provide fan users the best approaches in minimizing fan noise.

System Impedance

The area between inlet and outlet ports of a cabinet may account for 60% to 80% of the total system impedance. In addition, the greater the air flow, the higher the noise level. However, the higher the total system impedance, the more air flow is required to provide the necessary cooling. Therefore, system impedance must be reduced to the lowest possible level in order to keep fan noise to a minimum.

Flow Disturbance

Obstructions along the path of the flow of the turbulent air generates

noise. Thus, obstructions, especially in the critical inlet and outlet area, must be avoided to reduce noise level.

Fan Speed and Size

Since a high speed fan usually generates more noise than a low speed fan, the latter should be tried and used whenever possible. Very often, a larger, slower fan is quieter than a smaller, faster fan while delivering the same air flow.

Temperature Rise

Air flow is inversely proportional related to allowable temperature rise within a system. A small change in the allowable temperature rise leads to a significant change in air flow required. Therefore, if there is a little compromise to the limit imposed on allowable temperature rise, a much smaller amount of air flow will be required. As a result, fan noise is remarkably reduced.

Vibration

In some cases, a soft and flexible isolator like a grommet may help to avoid vibration transmission.

Voltage Variation

Voltage variation affects the level of acoustic noise. When a higher voltage is applied to the fan, more vibration and consequently more noise is generated due to the increased RPM.

Design Considerations

The design of every component of the fan affects the level of acoustic noise. Low noise levels can be achieved by accurate dimensioning of the winding core, the design of impeller blades, housing and by precision manufacturing and balance of bearings, if used.

(020108-1)

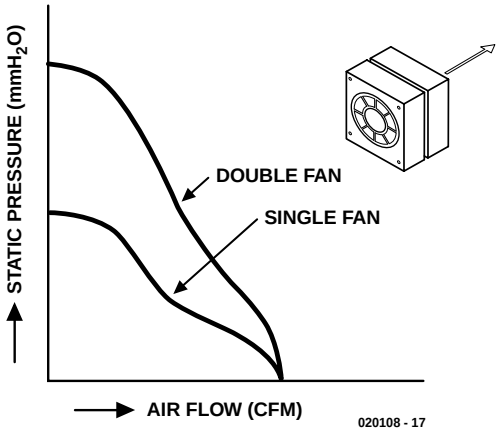


Figure 7. Performance of series-mounted fans vs. single fan.

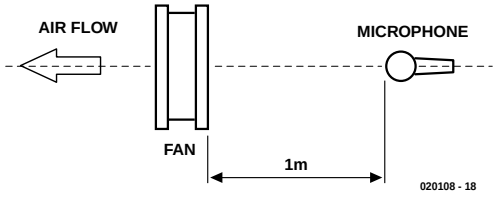


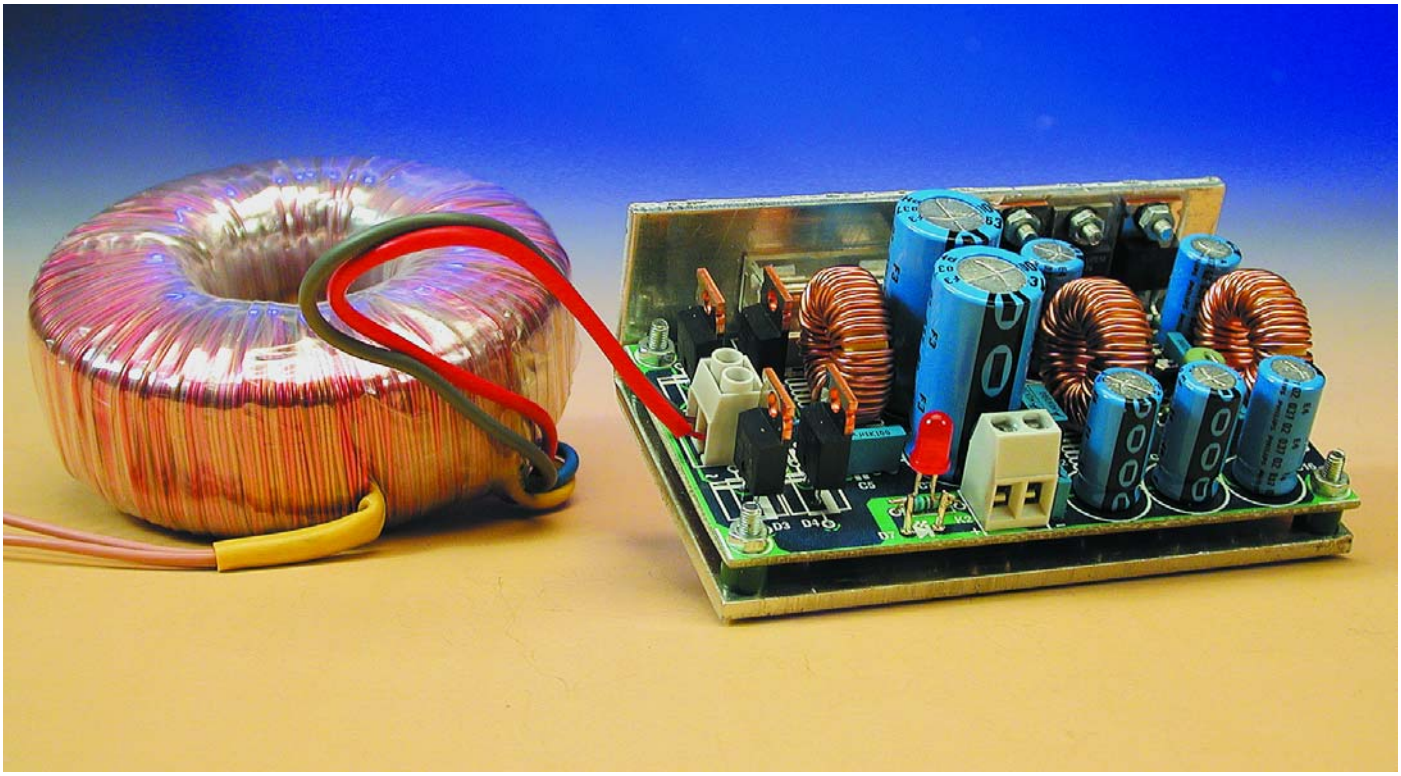
Figure 8. SPL measurement.

17 V/10 A Switch-Mode Power Supply

For the active multimedia subwoofer

design by T. Giesberts

This compact stabilised power supply was specifically designed to obtain the best performance from the recently described subwoofer that forms part of our Active Loudspeaker System. It produces the ideal supply voltage of exactly 17 V, is capable of sourcing up to 10 A and has an excellent efficiency.



In the article for the active subwoofer we already mentioned that there were two choices for the power supply: the usual combination of a transformer, bridge rectifier and smoothing capacitor, or a stabilised 17 V supply. We promised to publish a design for this in the near future, and here it is.

The first question many readers will ask is why we have used a stabilised power supply in this instance, which we normally don't do for power amplifiers. The main reason for this is that the TDA7374B power amplifier IC can only operate at a relatively low supply voltage. The best performance and maximum power output of this integrated power amplifier are obtained at a supply voltage of 17 V. This is a fairly unusual voltage, which would be difficult to achieve using standard components. The closest you could come to this is with a 12 V transformer, but with rectification the voltage drop across the diodes reduces the maximum voltage to about 15 to 15.5 V. When compared to a supply voltage of 17 V this reduces the maximum power output by a considerable amount, and that would be a pity. There is really only one solution when you want to obtain the maximum power output from the TDA7374B: a stabilised supply providing exactly 17 V.

The next point to consider is how such a stabilised supply should be implemented. A 'classic' linear design requires a heavy-duty regulator, which has an unavoidable voltage drop across it. And at a maximum current of 6 A this should be kept into consideration. This gives it two clear disadvantages. Firstly, the voltage drop gives the supply a low efficiency. Secondly, all losses will of course be converted into heat, leaving the regulator to dissipate a lot of power, which therefore requires a substantial heatsink. Not an ideal solution really.

If you require a reasonable efficient design that has to supply large currents (as we do here), that leaves only one realistic choice: a switch-mode supply. After ample considerations we decided to use an old favourite: the LT1074 made by Linear Technology.

Step-down regulator

The LT1074 is an integrated step-down switching regulator, which can source a healthy 5 A and furthermore requires very few external components. This IC has previously been used in the 'In-car SMPSU', published in the June 2001 issue.

Advantages of the LT1074 are its reliability, response time, as well as its built in protection against overloads and short circuits. The fact that it is a step-down regulator has the advantage that a mains transformer with any secondary voltage between 18 V and 30 V can be used to provide the 17 V output.

One problem is that the LT1074 can 'only' supply 5 A, whereas we need at least 6 A at full power. We have solved this by cleverly connecting two virtually identical LT1074 circuits in parallel, which is described next.

Powerful pair

A quick glance at the circuit in **Figure 1** shows without a doubt that there are indeed two LT1074's and that there are far fewer external components than expected. Of the two ICs, IC1 functions as a 'master regulator', whereas IC2 only springs into action when the output current rises above 5 A.

We now have a look at the standard com-

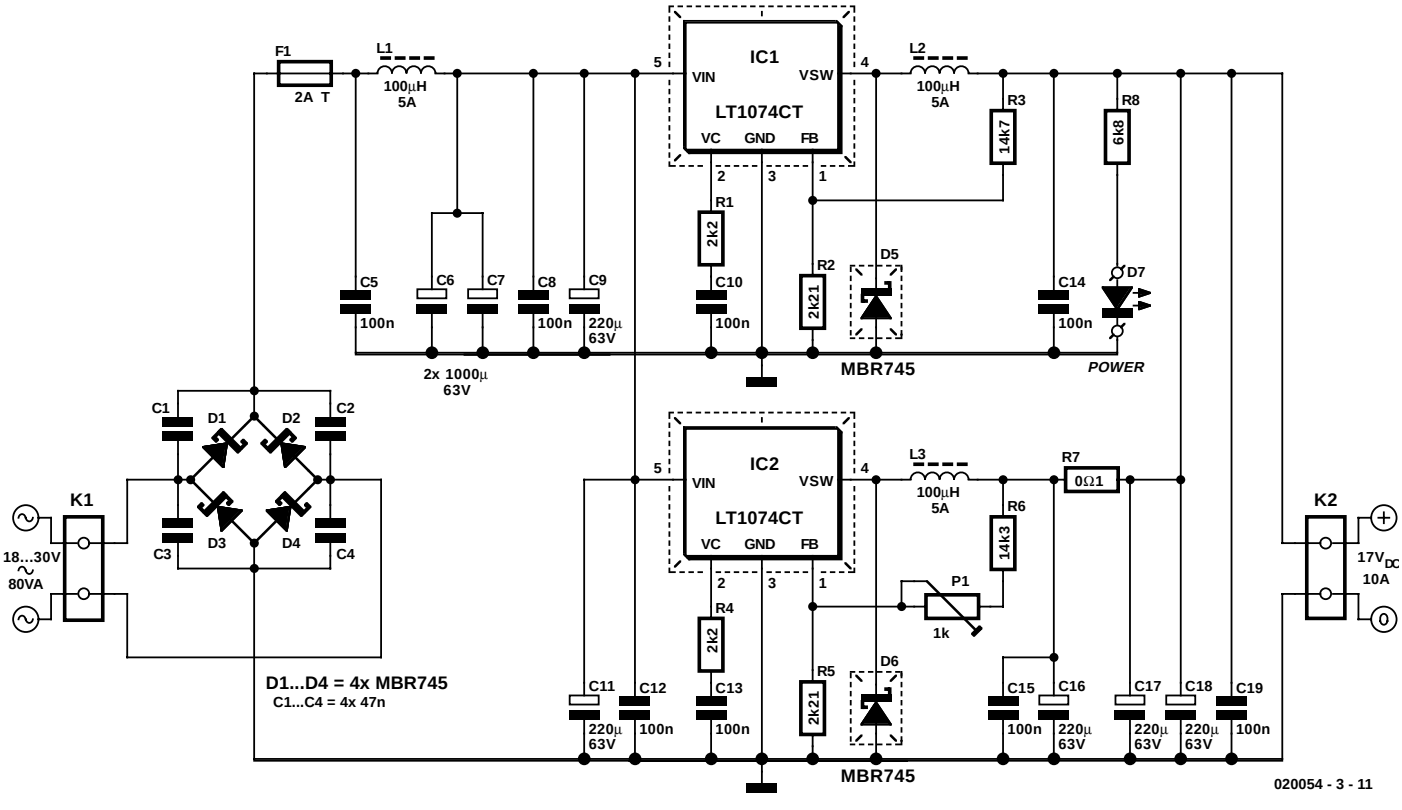


Figure 1. The 'beating hart' of the power supply is a pair of LT1074's.

ponents surrounding the ICs, initially limiting ourselves to the 'master regulator' (IC1), since this is used in a standard configuration.

The transformer voltage from K1 is first rectified using a discrete bridge rectifier (D1-D4). Capacitors C1-C4 have been added to suppress rectifier transients and other noise. The input voltage is then decoupled by L1 and C5/C8/C9, which also prevent switching pulses from the LT1074 finding their way back into the mains.

The output voltage is determined by resistor network R2/R3, where the voltage at pin 1 is compared to the internal reference voltage of 2.21 V. Since we've used a value of 2.21 kΩ for R2 (not coincidentally), you don't even need a calculator to determine that the theoretical output voltage is 16.91 V in this case. R3 has intentionally been connected to a point after inductor L2. Before L2 is a pulse width modulated signal, which has an average that corresponds to the output voltage, but which would throw the reference circuitry into utter confusion.

The R1/C10 network is required for frequency compensation in the LT1047. D5 functions as a freewheeling diode for inductor L2. The smoothing capacitors at the input and output (C6/C7 and C16/C17/C18) have intentionally been connected in parallel in order to reduce the size of the current pulses in individual capacitors. This gives them a longer life and also causes a reduction in the total ESR (equivalent series resistance) and parasitic inductance. With an eye on keeping the ESR as low as possible, a voltage rating of 63 V has been chosen for the electrolytic capacitors.

Now for the second LT1074 (IC2). This has actually been connected in parallel with the first and functions as a sort of 'emergency' supply; it will only start to contribute a current to the output when a heavy load causes the output voltage of IC1 to drop slightly. Two measures were taken to implement this. The first is the addition of series resistor R7 to the output. And secondly, the output voltage of IC2 can be varied using P1. The adjustment of this preset is very simple. With no load connected to the power supply, it should be turned clockwise to just *before* the point where the output voltage at K2 increases. The output voltage of IC2 is then just a fraction below that of IC1, leaving it in an almost dormant state until such time that the first becomes stressed, which is exactly what was intended.

Bridge rectifier and transformer

To keep the efficiency of the supply as high as possible we've used the same Schottky diodes for D1-D4 in the bridge rectifier as are used for D5 and D6 (the MBR745). These have a very low forward voltage drop, even at high currents, whereas this could easily rise up to 1.5 V for ordinary diodes.

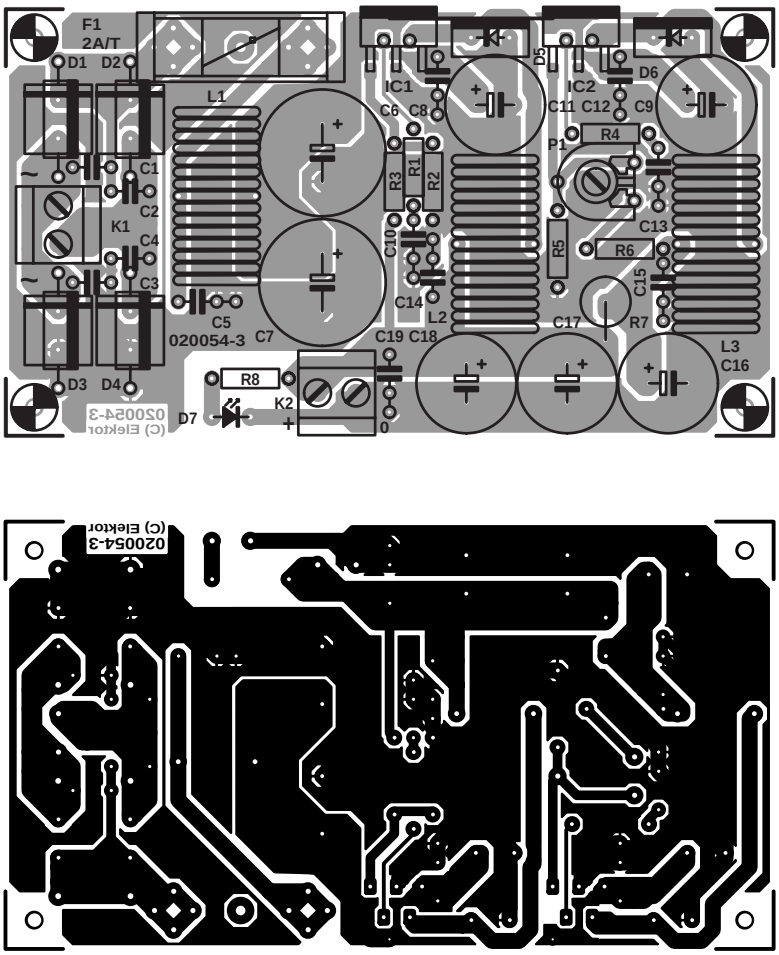


Figure 2. For a power supply that delivers a peak current of 10 A, the PCB turned out surprisingly small.

COMPONENTS LIST

Resistors:
R1,R4 = 2kΩ2
R2,R5 = 2kΩ21
R3 = 14kΩ7
R6 = 14kΩ3
R7 = 0Ω1 5W
R8 = 6kΩ8
P1 = 1kΩ preset

Inductors:
L1,L2,L3 = 100µH 5A, e.g., SFT12-50 (TDK)

Capacitors:
C1-C4 = 47nF
C5,C8,C10,C12-C15,C19 = 100nF MKT (metallised film)
C6,C7 = 1000µF 63V radial
C9,C11,C16,C17,C18 = 220µF 63V radial

Semiconductors:
D1-D6 = MBR745 (Schottky diode, 7.5A 45V piv)
D7 = LED, high-efficiency
IC1,IC2 = LT1074CT (Linear Technology)

Miscellaneous:
K1,K2 = 2-way CB terminal block, lead pitch 5mm
F1 = fuse, 2A/T (time lag), with PCB mount holder
PCB, order code **020054-3** (see Readers Services page)

There are still better Schottky diodes, such as the 20TQ045 made by IRF (also 45 V max.), which have an even lower forward voltage drop; these may obviously also be used. With an input voltage of 30 V and using MBR745 diodes, the efficiency was measured to be better than 85%. The quiescent current of the circuit is about 23 mA.

Keeping in mind the efficiency of the power amplifier and supply, we recommend that an 80 VA mains transformer is connected to K1. If you intend to use two subwoofer PCBs, you could use a single 160 VA toroidal transformer to supply two power supply boards (toroidal transformers usually come with two isolated secondaries). The same idea can also be used for the two satellites, which work very well with a rectified and smoothed supply. In this way there are several power supplies that are electrically isolated, thereby avoiding earth loops.

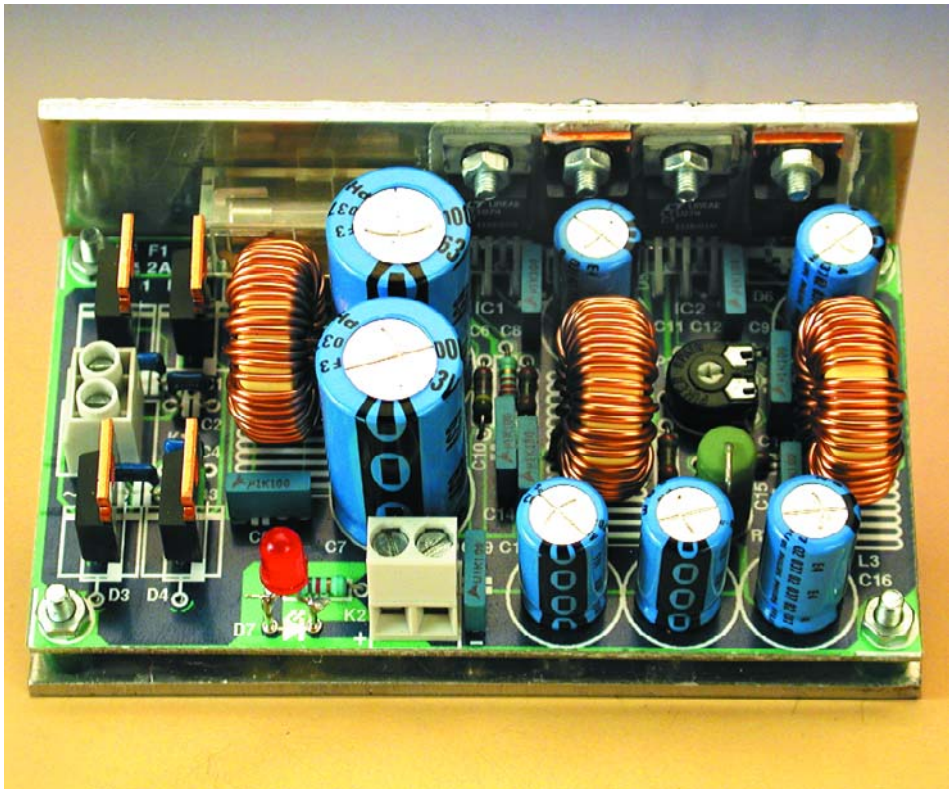


Figure 3. The PCB can be provided fairly easily with a homemade heatsink.

Construction

A very compact PCB has been designed for this circuit, which is shown in **Figure 2**. Populating this PCB shouldn't cause any problems as long as you keep to the usual sequence when soldering the components: the smallest come first, followed by progressively larger ones. Make sure that you don't forget the wire link that runs along the edge of the board behind the pins of IC1, D5 and IC2!

The three inductors, L1 to L3, are ordinary triac suppression chokes that are widely available. The inductance of 100 µH is not critical, but they should be rated for currents of at least 5 A.

The two regulators and their catch diodes have been placed near the edge of the PCB, making it easy to fit them to a heatsink. Its thermal resistance should be around 5 K/W and isolating washers should be used throughout. A small amount of thermal paste improves the thermal conductivity.

It is important that no mechanical strain is introduced to the leads of IC1, IC2, D5 and D6 when they are mounted to the heatsink (this could eventually cause the solder joints to fail). It is for this reason that the pads for D5 and D6 have been placed a bit further from the edge of the PCB; this introduces a small kink in the leads of the diodes, which reduces the strain on them.

Instead of a standard heatsink it is also possible to use a piece of aluminium with a thickness of about 3 mm, which is bent into an L-shape. The photo in **Figure 3** shows what we mean. The capacity of such a homemade heatsink appeared to be more than sufficient in practice, whilst it also made for a nice compact and robust module. The PCB is mounted to the heatsink using four nuts and bolts with stand-offs. These stand-offs have to be the plastic variety, since metal ones could cause shorts between the heatsink and the solder pads of D1 and D3 on the PCB. A bonus of this arrangement is that it can also provide some electrical shielding when the aluminium is connected via a solder tag and short length of wire to the ground of the circuit (the 'zero' connection of K2).

There are a few more practical points regarding the construction. There is space on the PCB to mount a set of normal axial diodes for D1-D4 in an R-6 package, such as the FR606. The efficiency of the circuit is reduced however, and the diodes become fairly warm. In order to prevent the inductors from vibrating and causing noises they should be glued to the PCB using an epoxy adhesive. LED D7 shows when the supply output is present and also functions as an on/off indicator; this LED should therefore be mounted in such a way that it remains visible once the circuit has been encased.

And as a final point we should mention the fuse: the rating of F1 may appear to be a bit low at first, but this value has been chosen to take account of the average power consumption of a subwoofer amplifier. The output current of 10 A should therefore be considered a maximum value and not a continuous current. Just to make things clear: the circuit is certainly capable of supplying a continuous 10 A, but the heatsink (and fuse rating) will have to be upgraded. The current circuit design is perfectly suited for a subwoofer supply.

When connecting the mains transformer to K1 it is advisable to include a mains fuse on the primary side. The exact rating will depend on the type of transformer used, but it will probably be around 200 mA(T).

(020054-3)

Speed Governor for Märklin Delta Control

one extra locomotive

Design by N. Körber DH0HAN

Three components and an arrow-style knob. This manual speed control for the Marklin DELTA multi-loc system adds independent control for an additional locomotive.

In multi-loc mode, Märklin's type 6604 Delta Control is capable of controlling four model locomotives. An external connection is available for a manual speed governor to control one additional locomotive. A suitable manual control is available commercially under the designation DELTA-Pilot, but do consider a 'home brew' circuit as shown here.

The circuit

As shown by the circuit diagram in **Figure 1**, only three components are required. The loc speed is effectively dependent on resistor R1 in combination with potentiometer P1. The locomotive does not move with the wiper of P1 turned to the 'top' end, i.e., with P1 effectively short-circuited. The locomotive may be reversed by pressing pushbutton S1.

Construction hints

The circuit does not require screening measures or special cables to connect to the 6604 Control Unit. As shown by the introductory photograph, the components are easily fitted in a small plastic (ABS) case. The photograph shows how the prototype was built.

If you decide to use a switch and a potentiometer with solder tags, the wiring may be 'in the air'. Resistor R1 should be a close-tolerance (1%) metal film type, or one whose value has been accurately established at 2,200 ohms.

The potentiometer in the circuit may have the (usual) tolerance of 10%.

The connections to the 6604 DELTA control unit are interchangeable.

More hints

The author happily employs the manual speed governor in combination with his DELTA Control unit, which has been in use since 1999. The printed circuit board in the control is labelled '39110 DELTA, Ver 3.4, 14/94'. The perfect operation of the manual speed governor was also proved beyond doubt by the Märklin Data Spy which will be published in a future issue. However, that does not imply that we can warrant proper operation of the speed governor with all available control units, or those that will be produced in the future.

(010108-1)

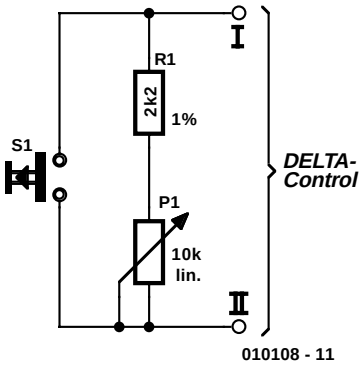


Figure 1. Circuit diagram of the Speed Governor.

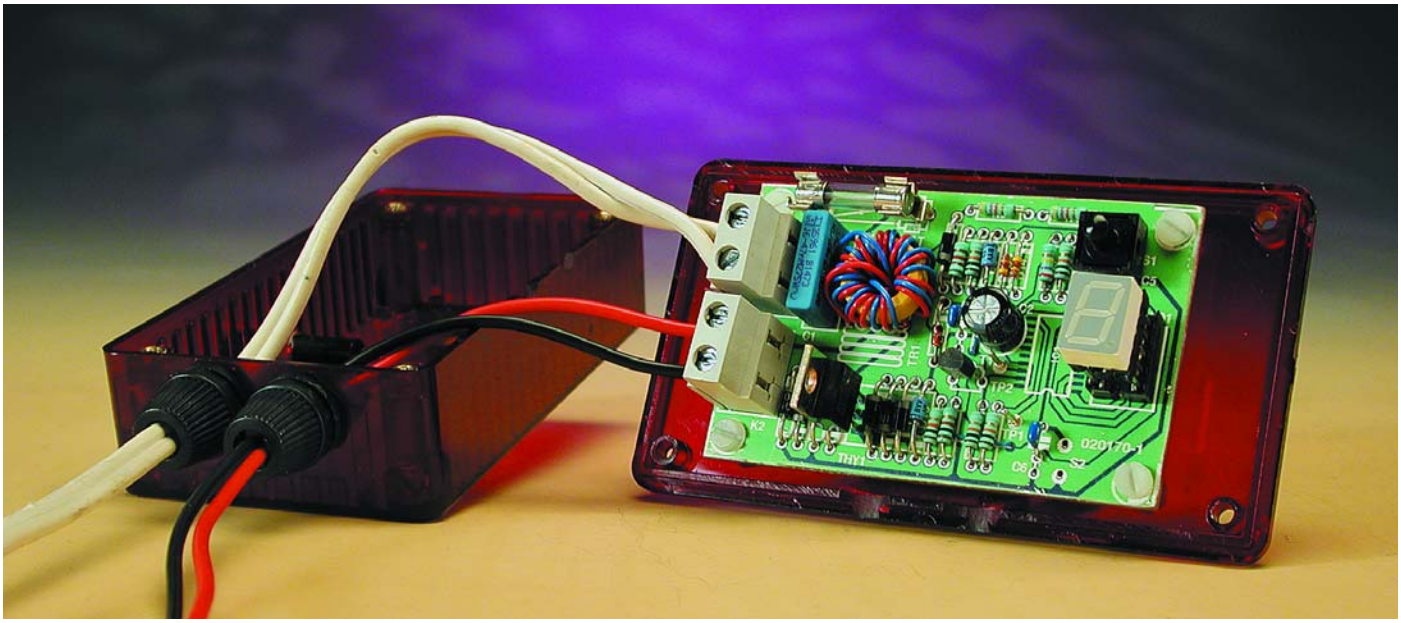
Author's homepage
www.koerber-home.de

Intelligent Fan Timer

Using the MSP430 low-power processor

Design by J. Volkering

This processor-driven automatic fan controller does not just run in parallel with the toilet light, but instead allows the user to adjust the running time of the fan using a single-button system. The selected time can be read from a single-digit display. The fan can also be stopped early using the same button.



Most toilet fans are switched on and off along with the toilet light, and some continue to run for a certain amount of time after the light is switched off. Since the need for fresh air is closely related to the particular reason for visiting the toilet, the author of this article wanted to have a bit more influence over how long the fan continued to run. The nicest solution would be to have a fan whose running time could be entirely chosen by the user. Naturally, switching the fan off early should also be possible, and it would be a useful extra feature to have the operating time of the fan shown on a display.

In itself, such a circuit is not particularly difficult to implement. However, when it is necessary to count down a preset time, control a display and drive a switching element, using a small microcontroller can easily prove to offer advantages. This also makes it possible to keep the operating controls simple. Here we have chosen a single-button system, which works very nicely in practice: you press the button once to switch on the fan, press it repeatedly to extend the time, and hold the button pressed to switch off the fan.

If a circuit of this sort is to be made as compact as possible, it is helpful if it can be powered directly from the mains without a transformer, using a simple voltage stabiliser, rectifier and series resistor. To avoid dissipating too much power in the series resistor, the circuit should not draw too much current, which means we need to look for a low-

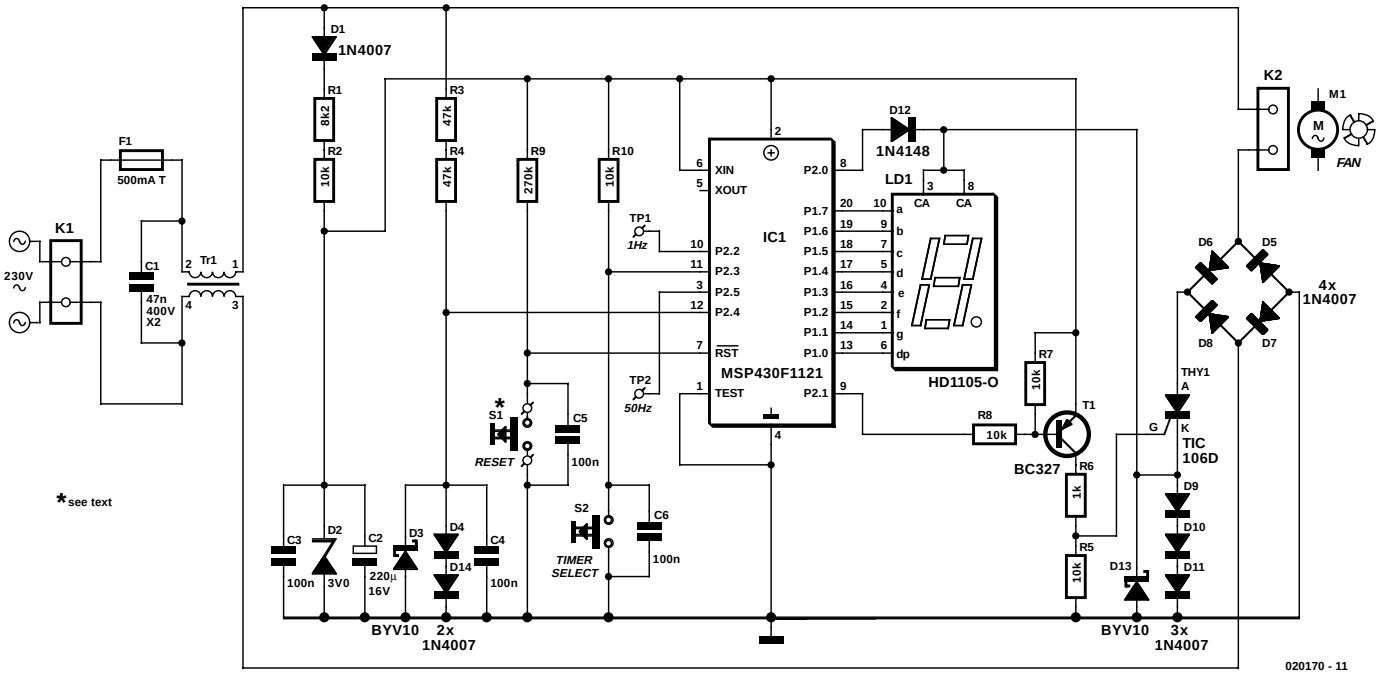


Figure 1. The schematic diagram of the timer, which is built around an MSP430F1121 microcontroller.

power processor. The Texas Instruments MSP430 series meets these economy requirements, and it has several other features that allow the number of components needed to be kept to an absolute minimum.

The MSP430 family

The Texas Instruments MSP430 family is a family of 16-bit RISC microcontrollers especially developed for low-power applications. Over the course of time, the family has been extended with versions having a mix of functions, such as A/D converters, comparators, timers, PWM outputs, watchdogs, LCD controllers, UARTs, synchronous serial ports and a multiplier. The on-chip memory can consist of a combination of RAM, ROM, one-time programmable (OTP) ROM, EPROM and Flash memory.

The MSP430 has several low-power modes. The current consumption decreases as more elements of the microcontroller are put into the low-power mode. In the fully active state, the current consumption is approximately 250 µA, while in the deepest low-power mode it is only 0.1 µA. Since we find 250 µA to already be very economical, we actually do not use the low-power modes in the fan timer circuit. The supply voltage ranges from 1.2 V to 5 V, depending on the type.

All members of the MSP430 family have an extensive oscillator circuit on board, with the choice of an internal oscillator, an inexpensive 32-kHz watch crystal or a crystal for the desired clock rate (8 MHz maximum). The frequency of the internal oscillator or digitally controlled oscillator (DCO) is adjustable, thus allowing the 50-Hz mains frequency to be used as a reference.

For the fan timer, we chose one of the least expensive types, the MSP430F1121. It has only 1 kbyte of Flash memory, 128 bytes of RAM, a comparator, a 16-bit timer with PWM outputs and a watchdog timer.

Most of the signal pins of the MSP430F1121 can be programmed as digital I/O pins according to the application, and each input can independently generate an interrupt. If the I/O ports are configured as outputs, the total output current can be as much as 48 mA. Other functions of the MSP430F1121, such as the comparator and the PWM outputs, can also be programmed to be connected to the output pins.

The MSP430F1121 has a boot loader in ROM for programming the Flash memory, even if the microcontroller is already soldered to the circuit board. The MSP430 family is programmed via a serial port, which is called a JTAG port, and the software is also tested using the same port. This port is shared with the general I/O pins.

More information and sample programs can be obtained from the Texas Instrument website at
http://focus.ti.com/docs/prod/folders/print/msp430f1121.html

Schematic diagram

Figure 1 shows the complete schematic diagram of the fan timer. As can be seen, it does not involve very much hardware.

The supply voltage is taken directly from the mains network via connector K1, fuse F1 and a noise-suppression choke (Tr1). Tr1 is wound bifilar in order to present a high impedance to noise pulses, which are typically asymmetric, while C1 is specifically included to provide a short circuit for undesired noise pulses.

After half-wave rectification by D1, the supply voltage is stabilised at a nominal level of 3 V using the series combination of R1, R2 and D2. The series resistance for the Zener diode has been intentionally been divided between two resistors in order to keep the voltage drop across each resistor within limits and allow standard 0.5-W resistors to be used. The average current through this series circuit is approximately 5 mA. The combined average dissipation of R1 and R2 is thus 0.5

W, which is the power consumption of the timer in the quiescent state.

Capacitor C2 has a relatively large value, since it must smooth the ripple in the supply voltage resulting from half-wave rectification as well as provide an energy buffer to allow the decimal point of the display to be illuminated every five seconds when the timer is quiescent.

The 50-Hz reference signal is generated by R3, R4, D3, D4, D14 and C4. Here again, the series resistance is divided between two resistors to limit the voltage across each resistor. Schottky diode D3 has been added to prevent the 50-Hz reference signal from going more than 250 mV below the ground level of the supply, in order to prevent damage to the MSP430. C4 suppresses any high-frequency noise that may be present at the 50-Hz reference input.

The load (the fan motor) is switched on and off by a thyristor (THY1) incorporated in a diode bridge formed by D5–D8. In effect, the thyristor shorts the plus and minus terminals of the bridge so that an alternating current can flow thorough the bridge.

When the thyristor is conducting, a portion of the full-wave rectified sine wave is present across the bridge, with a peak voltage of approximately 1.8 V. The supply voltage for the display is taken from this voltage. The advantage of this is that the display draws energy only when necessary, thus avoiding placing an extra load on the supply for the MSP430. Schottky diode D13 has also been added here to prevent the pins of the microcontroller from going more than 250 mV below the supply voltage.

The segments of the display can be energised by pulling the associated outputs P1.0–P1.7 low. The output resistance of the microcontroller ports, which is around 100 Ω, limits the current through the display segments to approximately 3 mA. However, the LED display should have a voltage drop of at least 1.5V across the LEDs, since otherwise external current-limiting resistors must be added. This was not necessary with the type shown in the components list. In order to obtain sufficient light intensity, it is recommended to choose a display with high optical efficiency at low currents. The selected type (HD1105-O) provides 21,000 μcandela at 10 mA.

At the zero crossing of the mains voltage, the thyristor will go off when there is no anode–cathode voltage and no gate voltage. However, if the gate voltage is held at 3V, it will immediately switch on again as soon as the voltage across the thyristor is greater than the peak on-state voltage (V_{tm}) of 1.7 V. This avoids switching spikes during the active state. A supplementary benefit is that there is no need for special timing software to synchronise the gate drive to the 50-Hz signal, except when it is first switched on.

In order to prevent voltage spikes that may be induced at the gate of the thyristor from disturbing the operation of the MSP430, the thyristor is driven via a transistor buffer stage using T1. This would not have been necessary just to supply the gate current, since the MSP430 has enough 'muscle' to drive the thyristor directly.

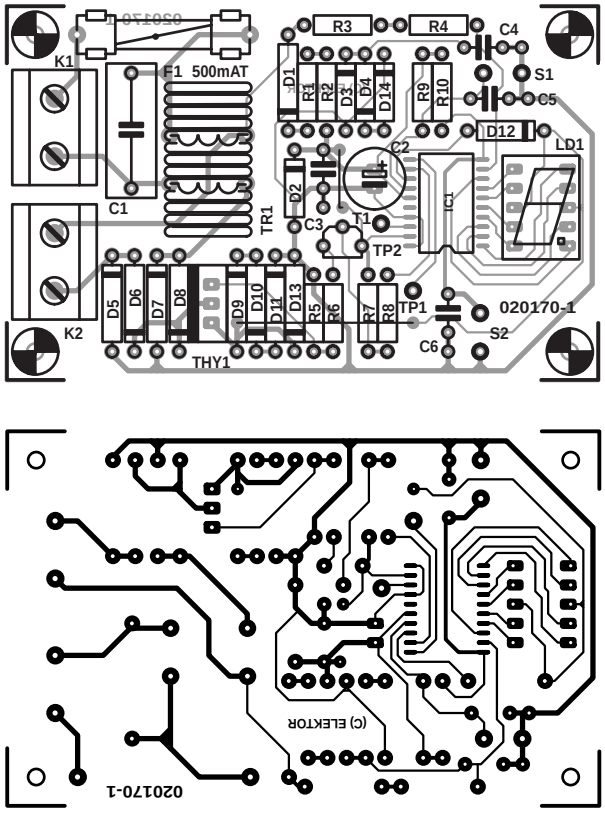


Figure 2. The track and component layouts for the timer printed circuit board.

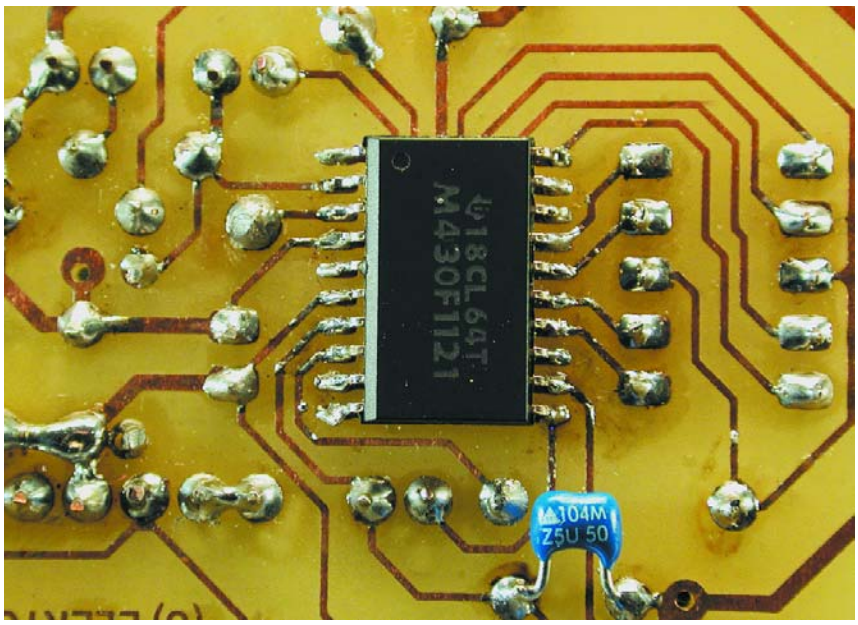


Figure 3. Fitting the MSP430 requires better than average soldering skills.

COMPONENTS LIST

Resistors:
R1 = 8kΩ2
R2,R5,R7,R8,R10 = 10kΩ
R3,R4 = 47kΩ
R6 = 1kΩ
R9 = 270kΩ

Capacitors:
C1 = 47nF 400V, Class X2
C2 = 220µF 16V radial
C3-C6 = 100nF

Semiconductors:
D1,D4-D11,D14 = 1N4007
D2 = zener 3 V/500 mW
D3,D13 = BYV10
D12 = 1N4148
T1 = BC327
THY1 = TIC106D
LD1 = HD1105-O
IC1 = MSP430F1121, programmed, order code **020170-41**

Miscellaneous:
K1,K2 = 2-way PCB terminal block, lead pitch 7.5mm
F1 = fuse, 500 mAT (time lag), with PCB mount holder
S1 = pushbutton, 1 make contact, Class 2 (for test purposes only)
S2 = pushbutton, 1 make contact, Class 2, e.g., Omron type A3DT-7111 with cap A3DT-500R (Farnell)
Tr1 = approx. 20 turns, bifilar, on Philips core TN14/9/5 (material: 3C85)
Enclosure: e.g., Hammond type 1591B (red)

Disk, project software, order code **020170-11** or Free Download

PCB available from **The PCBShop**

The MSP430 hardware

The display is connected to port P1.x, with the segments being illuminated with the associated outputs are low.

A reset circuit (R9, C5 and S1) has been added so that the microcontroller starts up in a well-defined state on power-up. The reset switch is only intended to be used for testing. Once the circuit has been fitted into an enclosure, switch S1 will anyhow no longer be accessible.

Pushbutton switch S2 is used to activate and stop the timer. If an I/O pin of the MSP430 is programmed as an input, the input signal passes through a Schmitt trigger, so a relatively slow input edge does not pose a problem for the MSP430. The contact bounce of the switch can thus be suppressed using a simple capacitor connected to the input, without requiring debouncing in software.

When the fan is not running, the internal 16-bit timer of the MSP430, which is driven by the internal oscillator, is used to indicate that the circuit is quiescent by energising the decimal point of the display for half a second every five seconds. This is done by setting port P2.0 high to apply a voltage to the anode of the display via D12. The decimal point of the display is then briefly pulled low by P1.0. When the fan is active, port P2.0 is held low to avoid loading the 3-V supply. Power for the display is then taken from D9–D11.

When S2 is pressed, the MSP430 switches from the quiescent state to the operational state. The hardware timer is switched off and the internal comparator is programmed to compare the 50-Hz input signal to 0.25 V_{cc} (750 mV). The output of the comparator is fed to port 2.5 (test point TP5). Every 20 ms, a comparator interrupt is triggered by a zero crossing of the mains voltage.

The entire housekeeping of the timer is derived from this 50-Hz interrupt, while the MSP processor is clocked by the internal oscillator (at approximately 1 MHz). First the display is energised, and on the next zero crossing, the thyristor is triggered into conduction.

The MSP430 software

We can start with the reassuring news that you can purchase a pre-programmed MSP430 chip via Readers Services. For those who like to do things themselves, the software and source code are also available on a diskette, or they can be downloaded from the *Elektor Electronics* website.

Now for a few details about the software. There are four software counters active in the MSP430. The first one runs in the interrupt service routine for the 50-Hz comparator and counts up to 50, in order to generate a 1-Hz signal (test point TP1). This 1-Hz signal in turn serves as a clock pulse for the second software counter, which counts up to 300. This is equivalent to 5 minutes, which is the minimum duration for which the timer can be activated. The 5-minute counter pulse, in turn, drives a software counter whose value matches the indication on the display. If S2 is pressed repeatedly, the displayed value can be increased up to a maximum of 9. This is equivalent to an activation time of 45 minutes. There is no use pressing S2 again after this value is reached, since 9 is the maximum.

If the timer must be switched off early, S2 must be pressed and held for more than three seconds. As soon as S2 is pressed, a fourth counter starts incrementing every second as long as S2 remains pressed. If S2 remains

Programming your own microcontroller

There is an inexpensive development tool available for trying out the MSP430, which is called the Flash Emulation Tool (FET). In practice, this tool also proves to be suitable for developing complete applications, such as the fan timer described in this article.
The FET (MSP-FET430x110; see www.ti.com) is supplied with a limited version of the IAR workbench development environment, called 'Kickstart'. It is limited in terms of the size of C programs accepted by the loader/debugger, which is 2000 lines after compilation. This amount of lines is certainly necessary, since the header file that is also supplied occupies nearly 1000 lines by itself, but it does give you descriptions of all the registers and bit positions of the entire processor. That saves quite a bit of work when you are writing your first program.
The code/debugging limitation does not apply if you work with assembler. For those who like to write in assembler, the MSP430 family is easy to program, since it has an orthogonal processor. This means that all instructions can use all possible types of addressing on all registers and RAM. Since the MAP430 has a RISC architecture, there are only 27 different basic instructions, although there are 7 different addressing options. The MSP430F1121 FET tool is connected directly to the printer port of a PC and costs approximately £35.

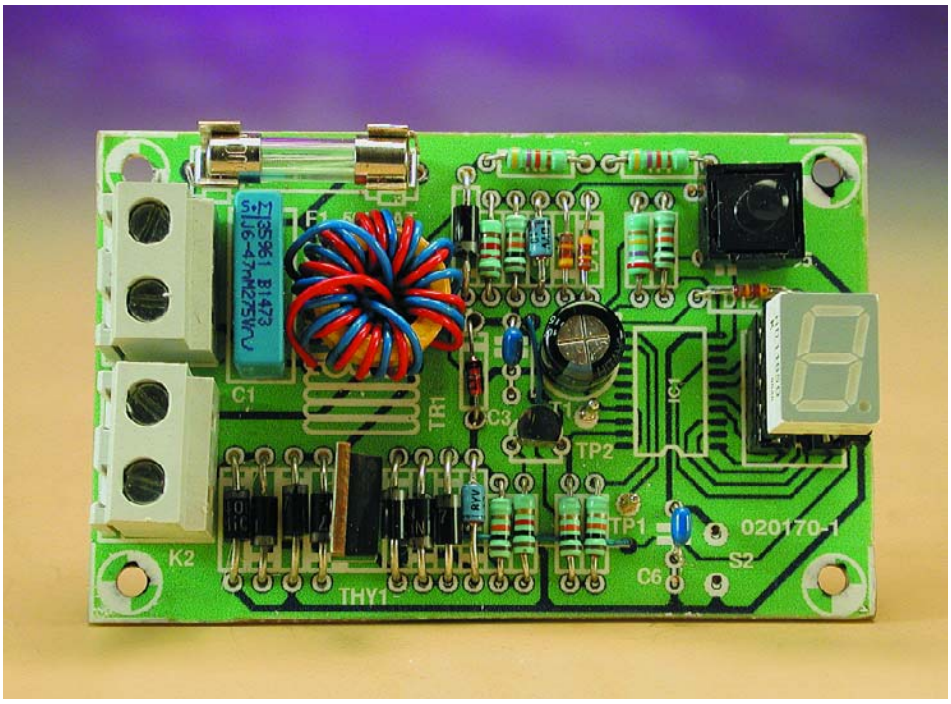


Figure 4. The component side of a fully assembled board.

pressed for longer than 3 seconds, this counter reaches a value of 3 and the thyristor is switched off.

The fan is switched on and off within the 50-Hz interrupt service routine, which is triggered on the zero crossings of the mains voltage. The timer thus always switches the fan on or off on a zero crossing.

When the timer switches off or is not active, the comparator and associated interrupt are disabled, and the internal hardware counter is again enabled in order flash the decimal point of the display to indicate that the circuit is in the quiescent state. The watchdog is continuously addressed during both the quiescent and active states, in order to prevent it from generating a system reset.

Practical implementation

The circuit board layout designed for the fan timer is shown in **Figure 2**. This circuit board is available from the PCB Shop (see the *Elektor Electronics* website).

There is only one difficulty with building the circuit board, which is fitting the MSP430 SMD IC. This little jewel is fitted to the solder side of the board, and it is a very good idea to do this right at the start.

First tin the solder pads on the board, and then clean them a bit using desoldering braid. Place the IC on the solder pads, and then attach two corner pins using a small soldering iron with a sharp point. After this, solder the remaining pins. Be frugal with the solder, in order to avoid shorts between the pins. Check to make sure that no undesired solder bridges have been formed and that all pins are securely soldered. Excessive solder can be removed using desoldering braid. A fitted SMD IC is shown in **Figure 3**.

After this, you can turn over the board and fit the 'normal' (leaded) components to the board. This is best done in the usual manner, working from low to high, starting with the wire link next to C3 and finishing with the relatively large circuit-board terminal strips K1 and K2.

The noise suppression choke Tr1 can easily be wound on a small (14 mm diameter) toroidal core (Philips type TN14/9/5 with 3C85 material). Take two lengths of insulated installation wire with a diameter of approximately 0.5 mm and wind them around the core approximately 20 times. The ends of the wire at the beginning of the windings are terminals 1 and 3 in the schematic diagram, while the other ends are terminals 2 and 4.

The display should be mounted on a socket, possibly elevated somewhat by using two 4-way pin headers. Pay careful attention to the polarity of the diodes, since a mistake can result in damage to the microcontroller. The component side of a correctly build board is shown in **Figure 4**.

After all components have been soldered in place, a seal coat should be applied to the solder side. An inexpensive way to do this is to use transparent nail polish (without glitter!).

After this, the board can be fitted into an enclosure. With regard to electrical safety, only plastic enclosures are suitable. The prototype was fitted into a transparent red plastic Hammond enclosure. This has the advantage that there is no need to saw an opening for the display, since the displayed number can be read through the plastic. If you do make an opening in the enclosure for the display, it must be covered by a piece of transparent plastic for safety. With a transparent enclosure, the only openings needed are a hole for the connection wires and a mounting hole for pushbutton switch S2. Be sure to use the switch shown in the components list for S2, and use plastic screws to attach the circuit board to the enclosure.

When testing and installing the circuit, remember that it is electrically connected to the mains network! Always disconnect the mains leads from K1 before working on the board, and before you start, read the Safety Information page published from time to time in *Elektor Electronics*. If you want to do things fully by the book, attach an identification label to the bottom of the enclosure once you are done, showing the project number and the value of the fuse.

Conclusion

The amount of power this circuit can switch depends on the diode bridge (D5–DS8) and the thyristor (THY1). The types shown in the components list can handle 1 A. If you want to switch larger currents, you will have to use correspondingly higher-power components. Note that the thyristor must be a type that triggers with a minimum gate voltage of 2 V at a gate current of less than 100 µA, since the drive current for the thyristor is provided by the 3-V supply.

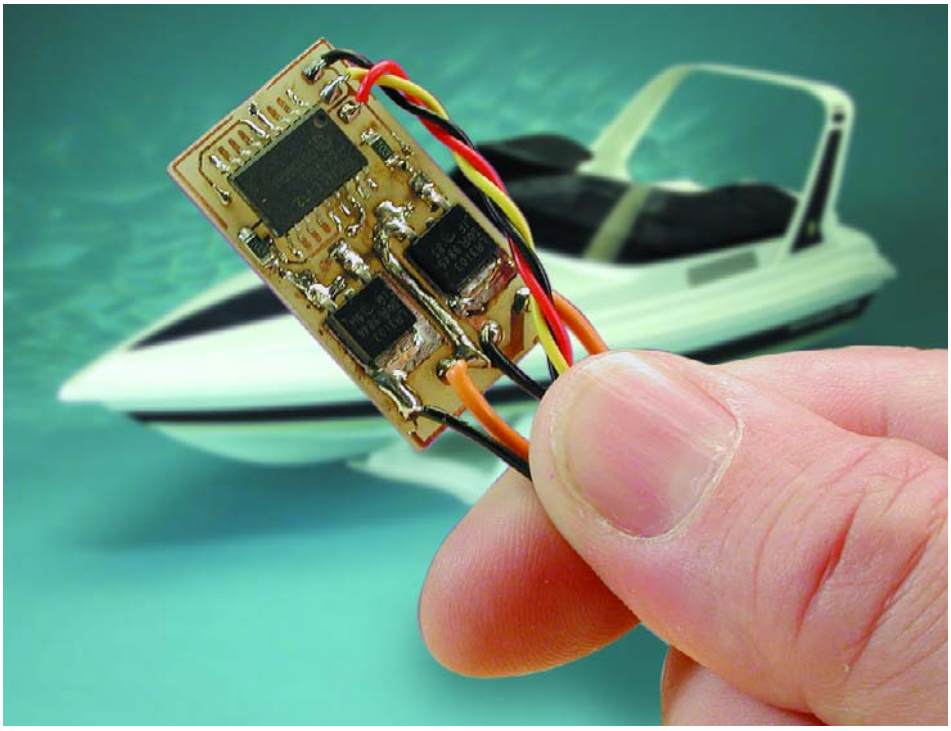
(020170-1)

Add-on Switch Unit for R/C Models

Provides two switches at 30 A

Design by A. Pistorius

This switch unit is designed to drive two extra functions in a model via a switch-channel on the transmitter. A soft start function is provided on one of the channels for turning on electrical motors smoothly. The correct control signal and the supply are derived from the receiver in the model.



Radio controlled boats become a lot more interesting when, apart from the standard control functions, a few other functions can be controlled remotely. Most of you will be able to think of a few examples that would be useful. The simplest implementation is with a couple of micro switches that are actuated by a servo.

An alternative switch unit can be built using a monostable multivibrator (MMV) and a flip-flop. The MMV compares the incoming pulse width modulated signal with a reference pulse width (about 1.5 milliseconds). When the pulse is longer than 1.75 milliseconds, the flip-flop is triggered and the output becomes active. If this isn't the case then the MMV is reset. A second function can be similarly implemented by detecting pulses that are shorter than 1.25 milliseconds.

A suitable combination would be the 74LS123 (dual retriggerable MMV) and the 74LS73 (dual JK flip-flop), which would provide two extra functions. Standard CMOS components can also be used. Such a circuit can be made fairly small through the use of SMD components.

Switch units that are based on this principle are commercially available (such as the 'Duoswitch' made by Robbe), but they aren't cheap. Home construction is therefore recommended, using for example a popular servo IC, the ZN409 (or ZN419), which was used in the 'speed controller for model train' (*Elektor Electronics* December 1999). Unfortunately, this IC is no longer obtainable, so we'll have to think of a different way to tackle this problem.

For an alternative device that can be used in a switch unit you need look no further than a PIC microcontroller, such as the popular PIC16C712, made by Microchip. Usually the PIC16F84 is used in home construction circuits, but in this case we have specifically chosen the 16C712 because it includes PWM outputs. This happens to be the first time that this chip is used in an *Elektor*

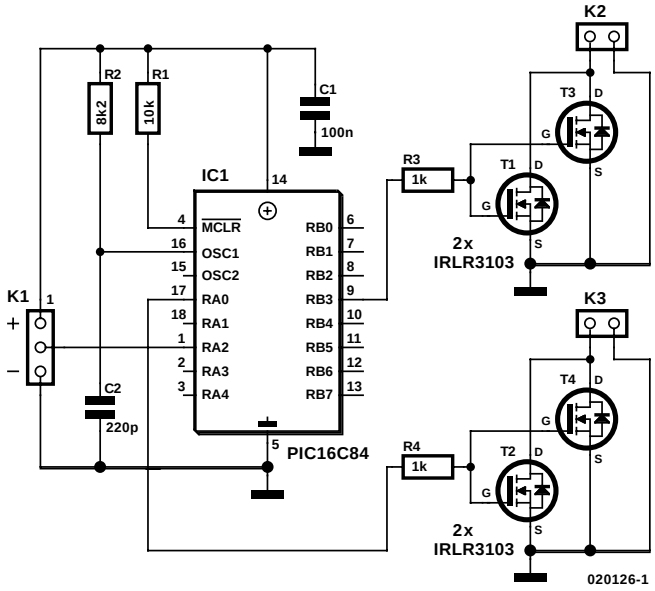


Figure 1. The circuit diagram of the switch unit.

Electronics circuit.

After this introduction the following description of the circuit won't come as a surprise: it has an input for the pulse width modulated signal and two outputs, which each drive an output stage with one or two FETs. An inexpensive RC clock generator completes the circuit. In this application there is no need to use a quartz crystal to obtain a very accurate clock frequency. As with many micro controller based circuits, the amount of soldering required is (sadly?) very limited.

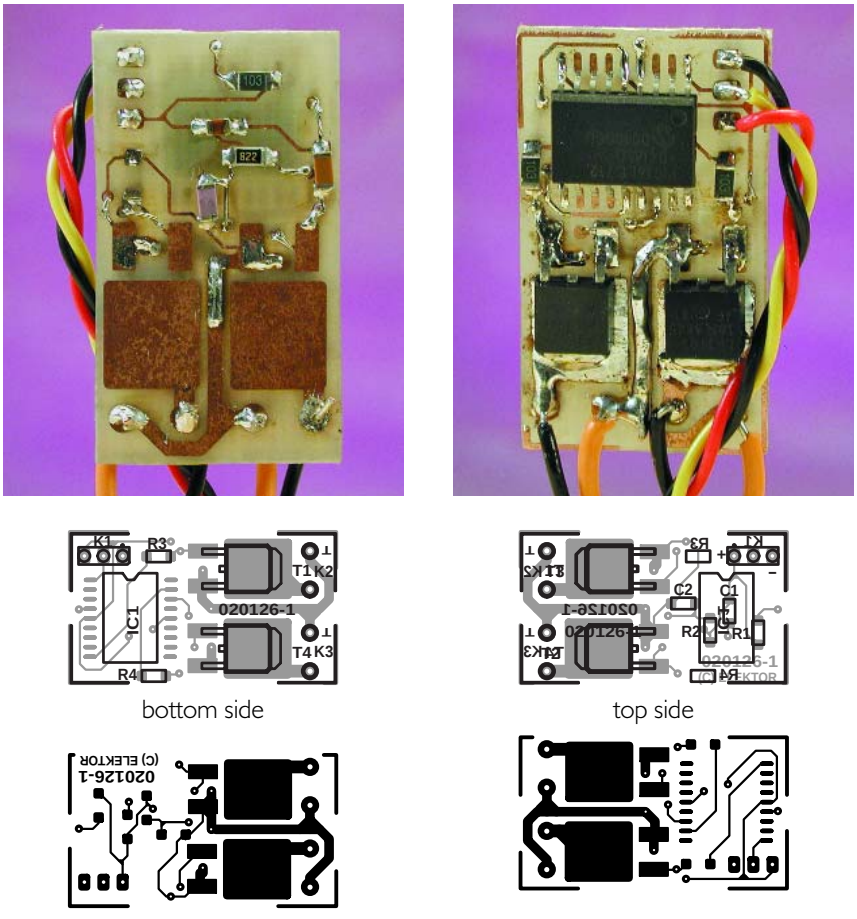


Figure 2. Populating the PCB for this circuit is a routine task.

The circuit diagram

The circuit diagram of the switch unit shown in **Figure** 1 is conspicuous by its simplicity. The hart of the circuit is obviously IC1. This controller receives the servo signal from the receiver via K1. Outputs RB3 and RA0 each drive two FETs via a resistor. The FETs are connected to the outside world via K2 and K3.

The FETs used here are each capable of switching 20 A. This current is limited by the packaging, since the device itself is capable of switching a current up to 50 A! In order to provide a safety margin, it is recommended that the switched current is kept a bit below the stated 20 A. A value of 15 A per FET is more sensible. Since we're using two FETs per channel, each channel can deliver a continuous 30 A, and that's a substantial value! When a current of 15 A or less is required, either one of the FETs can be left out.

The RC network R2/C2 determines the operating frequency of the controller. The values shown here give a clock frequency of 400 kHz ±10%. The exact frequency isn't very important as the software still functions properly when the frequency deviates by a few percent.

R1 keeps the reset input of the controller at a logic High level and C1 prevents any interference produced by the controller from reaching the receiver. The circuit is powered via connector K1, which is connected to the receiver.

Figure 2 shows the printed circuit board for the switch unit. For modellers who have some experience with SMD components, the construction will be a piece of cake. The PIC16C712 can be obtained ready-programmed from the *Elektor Electronics* Readers' Services.

The software

The circuit diagram alone is not sufficient to explain the workings of the circuit. The PIC16C712 cannot function without a program. This also shows how powerful a microcontroller is compared to a dedicated IC such as the ZN409 that was mentioned earlier. Any restrictions of these types of IC can easily be overcome with a microcontroller. One of the disadvantages of the ZN409 was that there was no provision for a soft start output, without using extra components. With the use of a microcontroller these types of function can be implemented easily. This program uses one of the outputs (K2) as a soft start (and soft stop) output. This means that the switch-on doesn't happen abruptly, but is gradually increased using a PWM signal over a period of about 1 second. The switch-off also happens in a gradual manner. This method of switching is par-

ticularly useful when an electrical motor is driven by the output. With a normal switch the motor will start abruptly. This obviously creates a large mechanical stress on the fixings of the motor. When the motor is provided with reduction gearing then this stress becomes much greater and could possibly lead to damage to the gearing and/or the fixings.

The software has been set up as a pulse width detector, using a counter that measures the period of a pulse at the input. The largest expected pulse width is about 2 milliseconds. If we use 200 instruction cycles for this then we can determine the pulse width to a high resolution. Furthermore, this counter value fits nicely in a single byte, keeping the program simple. This corresponds to a cycle time of 10 microseconds. The clock frequency required for this is 400 kHz. This frequency is obtained with a capacitor of 300 pF (in the prototype a capacitor was used with a tolerance of 5%) and a resistor of 6.2 kΩ. Of course it is possible to use different combinations as well, as long as the capacitor is greater than 20 pF and the resistor is between 3 kΩ and 100 kΩ (as recommended by Microchip). When an RC circuit is used for the clock generator the clock frequency (divided by 4) can be measured on pin 15 (OSC2/CLKOUT).

COMPONENTS LIST

Resistors:
R1 = 10kΩ
R2 = 8kΩ2
R3,R4 = 1kΩ

Capacitors:
C1 = 100nF
C2 = 220pF

Semiconductors:
IC1 = PIC16C712-041/SO, programmed, order code **020126-41**
T1-T4 = IRLR3101 (e.g., Farnell # 738-372)

Miscellaneous:
Servo cable
PCB, order code **020126-1** (see Readers Services page)
Disk, hex and source code files, order code **020126-11** or Free Download

And finally

Experimenting with microcontrollers completely changes the development cycle in modelling electronics. You need only modify the program to change the behaviour of the circuit. Those of you who have experience in programming PIC controllers in assembly language can download the source code from the *Elektor Electronics* website and modify it to your own needs. The source code includes sufficient comments to make its logic easily understood.

As can be seen from a recent article on modelling circuits (*Elektor Electronics*, February 2002) there is more than one road that leads to Rome. The circuit described here for the PIC 16C712 can again form the basis of various combined circuits (four inputs on PORTA and eight outputs on PORTB, with or without memory) and illustrates the simplicity and effectiveness of microcontroller circuits.

(020126)

Hot Air Soldering

Soldering without bits?

By J. Menke

While working for a cellphone manufacturer in the US of A, one of our readers picked up a neat trick using a hot-air gun and some surface-mount ICs.

In a busy development department of an industrial electronics company it just takes too much time fitting surface-mount components to PCBs using a conventional soldering iron. The tool of choice here is the hot-air soldering gun but you need to adjust your soldering techniques to take advantage of this device. After much practice, the author found the following procedure to be both simple and quick when it comes to mounting SMD packaged integrated circuits onto PCBs.

More than just hot air

The PCB pads must first be tinned by melting a small amount of solder onto each pad. The PCB manufacturer will already have pre-tinned the board but you need to add a little extra. The amount that you add now will form the finished joint between each component lead and pad, so try to make an equal dome of solder on all the pads.

Check the orientation of the component to the Layout (pin 1 to pad 1) and position the component as shown in **Figure 1**. Place a small dab of flux onto each pad and lead. Now with a hot-air gun set to approximately 290°C adjust the air flow rate to a gentle stream and with the hot air nozzle at a distance of between 5 and 15 mm, direct the stream of hot air over the component leads and pads so that they are evenly heated and no localised hot-spots are formed.

As soon as you are sure that all the pads and leads are up to temperature (the solder takes on a bright shiny mercury-like appearance), quickly slide the component onto the pads using tweezers. Alternatively tap the PCB with the tweezers so that the component slides onto the pads. As the molten solder touches the leads, it will magically draw the component into its correct position and centre it neatly on the pads.

The entire process should not take any longer than 20 seconds. If you are not successful after this time, remove the gun and wait for everything to cool down before cleaning, re-fluxing and trying again. Prolonged heating risks permanent damage to the component and PCB.

As with all surface-mount work, this method needs a very steady hand and some practice to avoid solder bridges between pads but a competent technician using this technique should be able to solder at a rate of around 100 leads every 10 to 20 seconds, quite a bit faster than automated placement machines!

(031004-1)

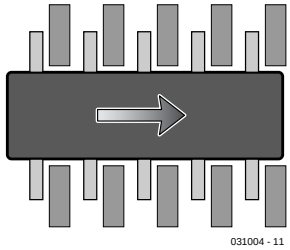


Figure 1. Position the component leads between the pads and slide into position as the solder flows.

RGB Colour Simulator/Indicator

With microprocessor control

By A. Steiger steigeralex@gmx.de

This prize-winning circuit from our Flash Micro Competition responds to a drive signal containing RGB (red/green/blue) values by making a multi-colour LED produce the desired hue. The circuit doubles as a converter between colour coding systems.

This entry received for our *Flash Micro Competition* (see the January 2003 issue) demonstrates that complex peripherals like a D/A converter are not really necessary when it comes to controlling a three-colour LED. Here, a microcontroller from the 80Cxx series handles all functions necessary to enable such an LED to produce all possible colours.

Any individual colour may be produced by mixing certain levels of three basic colours. There exist several systems to encode colours using different basic colours:

RGB (red, green, blue)
Raw data, range 0-255
(additive colour mixing)
RGB (red, green, blue)
Level, percentage 0 – 100%
CMY (cyan, magenta, yellow)
Level, percentage 0 – 100%
(subtractive colour mixing)
HSB (hue, saturation, brightness)
a model adapted to human perception.

Due to memory limitations, only these four colour coding systems are supported. Provided a memory extension is available, other models like CMYK, YUV, VCbCr and YIQ may also be implemented. This only requires a conversion from RGB code and vice versa.

User interface

The user interface consists of a keyboard with four pushbuttons to navigate through the menus and enter values, and a two-line LC display to indicate values.

The four pushbutton are on port lines P1.4 through P1.7, their states being treated as inverted in the program (on = 0, off = 1). Because the microcontroller already has pull-up resistors on its port line pins, the pushbuttons need only switch to ground.

The software assigns the following functions to the pushbuttons

Swx	Port pin	Function
Sw1	P1.4	Select colour coding system
Sw2	P1.5	Select components
Sw3	P1.6	Increment value
Sw4	P1.7	Decrement value

The circuit itself is marginal and consists of four driver stages for each LED element. The four LEDs occupy ports 1.0 through 1.3. The RGB LED contains one red, one green and two blue LEDs in a single enclosure. The component should be of the **diffuse** type, because in a multi-colour LED with a fully transparent encapsulation the individual colours are visible and the desired optical mixing effect does not occur.

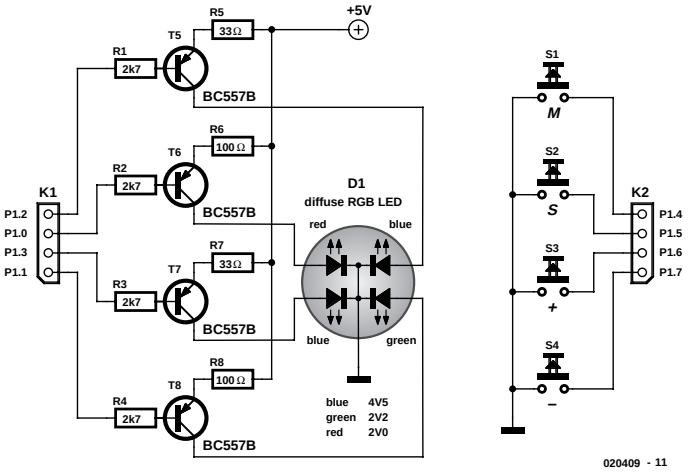


Figure 1. Circuit diagram of the simulator/indicator.

After the welcome text you are allowed to select the colour coding system and then enter the desired values using the relevant units and ranges (**Figure 2**). An interesting detail: if you enter the values for a particular colour model and then press Sw1 to select another coding system, the values are automatically converted.

Programming language

The use of BASIC is ruled out for this project because the brightness of individual LEDs is controlled directly by the microcontroller using pulsewidth modulation (PWM). The alternative at the other end of the spectrum, assembly language, proved far too cumbersome, so the author went for 'C' using the **Keil C51** compiler. An evaluation version of this industry-standard software is available free of charge (for non-commercial applications only) from the Keil website at www.keil.com. This software was also mentioned in relation to other projects and articles in *Elektor Electronics*. Apart from a few details, it is identical in syntax to the **Reads51** compiler from Rigel Corp., although it must be mentioned that the evaluation version of Reads51 has a code size limitation of 2 Kbytes. Mind you, that's the size of the executable (object) code, not the program code (the size of the RGB control program is about 3,300 bytes). Moreover, there is no floating comma arithmetic and it is not possible to create an LIB. Reads51 does not allow simultaneous control of the four LEDs and the LCD display. Also, Reads51 is too slow and generates too large object code files compared to C51. Based on the author's experience, the compactness of the object code created by C51 even beats many attempts at assembly-code programming.

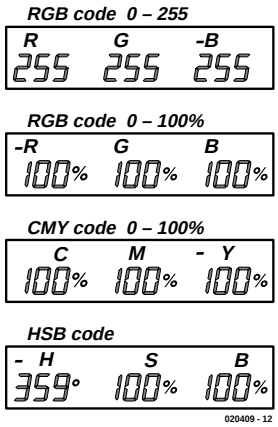


Figure 2. Value entry using pushbuttons.

The program

The pulse length is determined by a timer. Depending on the values assigned to Red, Green and Blue (0 – 255) , the requisite on/off time is continuously updated in the IRQ routine. Therefore, the four individual LEDs need to be controlled quasi-simultaneously 'in the background' while the foreground task entails scanning the keys and driving the LCD. The practical realisation of this daunting bit of software may be gleaned from the source code file, which is available from this month's Free Downloads section on the *Elektor Electronics* website. Look for zip file **020409-11**.

By transposing the software to controllers with a lager memory, further colour coding systems like CMYK, YUV, YcbCr, YIQ, CIE, Lab, and HSI may be implemented, as well as brightness control of the individual LEDs.

(020409-1)

yEnc & Co.

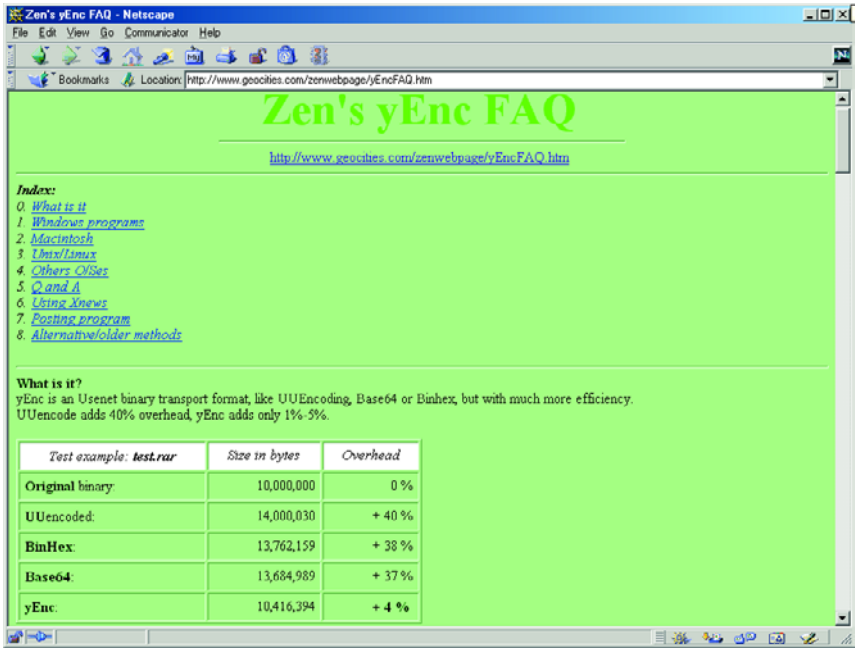
new encoding methods for attachments

By Harry Baggen

Although most computer users will never see anything else than professionally laid out websites and the odd email from friends, relatives and spammers, the global communication medium called the Internet has far more to offer. For example, there are tens of thousands of on line newsgroups that allow people with the same hobby or interest to exchange ideas (and flame one another!). Within these groups, a number of file encoding methods are used with strange names like yEnc and uuencode.

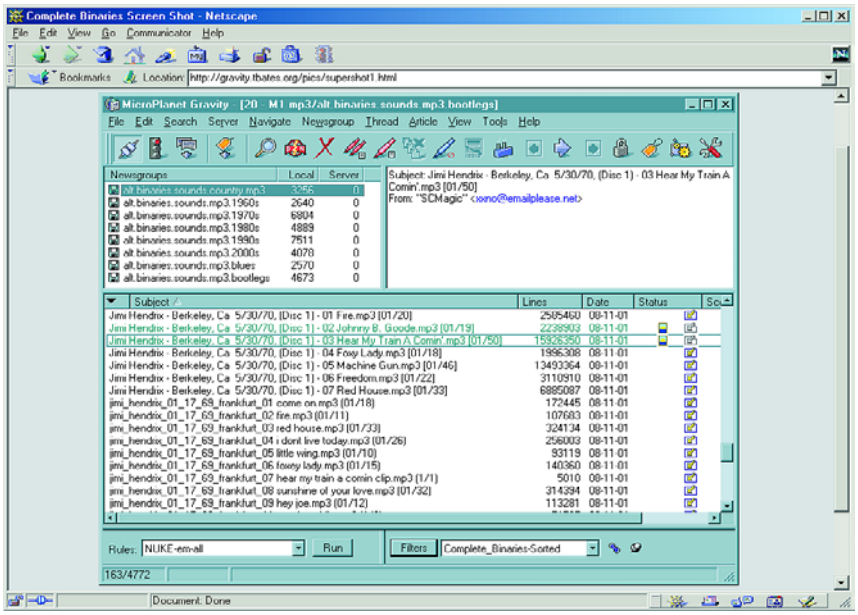
Not so long ago, you had to dial into a BBS (bulletin board system) to receive or send messages, or download files. In those early days many BBS users spent hours on end downloading programs and pictures, all adding considerably to the monthly telephone bill.

Today, the functionality of the former BBSs has been taken over by Internet newsgroups operating within a structure called Usenet (for User's Network). In the early days of the Internet, Usenet was a sub-section specifically designed to spread messages among larger groups of people. However, the creators of this system did not (and could not) foresee that Usenet users would want to send large binary files (or 'binaries'), too. Binaries are often used to convey programs, pictures and even music. Unfortunately, owing to lack of standardisation, several methods surfaced and were adopted to enable file attachments with Usenet messages.



Email and Usenet servers both use a fairly simple protocol. With email, the protocol is called SMTP (simple mail transfer protocol), while Usenet employs NNTP (network news transfer protocol). A common feature of these protocols is that they can only handle ASCII text, which seems to rule out the possibility of sending, say, an executable program in the form of an attachment with the message. Soon, however, ways were found to convert an binary into ASCII text and then send it on to Usenet. At the receiving side, the 'text' file is converted back into binary again and hey presto the receiver has a complete program. Various methods are available to do this, but the best known are Uuencode/Uudecode. Besides this famous pair you may come across methods like BinHex and Base64.

Today's Internet browsers have



an integrated newsreader utility ('Outlook Newsreader' in Microsoft Internet Explorer and 'Messenger' in Netscape). All encoding/decoding methods being integrated into these newsreaders, their operation is invisible to the average user. An attachment will automatically appear in its original form with the message, requiring no user action whatsoever.

Recently, a new file encoding method called **yEnc** [1], has popped up that does not seem to be supported by all newsreaders yet. So why introduce a new file encoding system when we are perfectly happy with the existing methods? Well, the latter suffer from a couple of distinct disadvantages, including inefficient operation. Due to the encoding process, a binary file will grow about 40% in size when sent as an email attachment. Consequently, sending such an attachment takes far more time than would be necessary if there was a means to convey the original. The overhead is far smaller, actually just a few per cent, when using the yEnc system (see **Zen's yEnc FAQ** [2]). Moreover, yEnc contains a built-in error correction system.

Besides positive response from Usenet users, critical notes also surfaced regarding yEnc (see **Jeremy Nixon's** article [3]), mainly because some minor shortcomings were discovered. In spite of these criticisms, yEnc did not take long to gain acceptance and popularity in the newsgroups.

The newsreaders that come with the major Internet browsers do not yet have yEnc support. If you try to download a yEnc file, you will only see a lot of garbled text on your screen.

Fortunately there are ways to overcome this problem. A separate newsreader that does have yEnc support could be used, while there are also plug-ins to upgrade existing readers. One of these plug-ins is called **yProxy [4]** from Brawny Lads. This little utility acts as a local proxy server for the existing browser.

Good, free, newsreaders with yEnc support include **Xnews** [5] and **Super Gravity** [6]. If you have some cash to spend, you may also consider buying **Forté Agent** [7]. The free version, called Free Agent, currently has no yEnc support.

A word of warning to those who think that it is now possible to download 'everything' from newsgroups: many Internet providers have banned newsgroups that allow 'binaries with messages', from their servers because of the tremendous increase in data traffic. Usenet is currently home to about 50,000 newsgroups that generate a daily traffic of the order of hundreds of gigabytes, requiring server capacities of 1 terabyte and more. If you are specifically interested in these attachments, then you will need to take out a (paid) subscription to a specialized news provider, or start a search for free news servers. Good search engines for this purpose include **Newzbot** [8] and **FreeNews** [9].

(035021-1)

Internet Addresses

[1] yEnc: www.yenc.org/

[2] Zen's yEnc FAQ: www.geocities.com/zenwebpage/yEncFAQ.htm

[3] Criticisms on yEnc: www.exit109.com/~jeremy/news/yenc.html

[4] yProxy: www.brawnylads.com/software/

[5] Xnews: http://xnews.newsguy.com/

[6] Super Gravity: http://gravity.tbates.org/super.html

[7] Forté Agent: www.forteinc.com/agent/index.php

[8] Newzbot: www.newzbot.com/

[9] FreeNews: http://freenews.maxbaud.net/